Using Head Position to Enhance Human Interfaces

by

Joel Wachman

B.A., Linguistics
Harvard University
Cambridge, Massachusetts
1983

SUBMITTED TO THE MEDIA ARTS AND SCIENCES SECTION, SCHOOL OF ARCHITECTURE AND PLANNING, IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF

MASTER OF SCIENCE

AT THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1990

Signature of the Author

Joel Wachman
Media Arts and Sciences Section
May 11, 1990

Certified by

Walter Bender
Principle Research Scientist
Thesis Supervisor

Accepted By

Stephen A. Benton
Chairman
Departmental Committee on Graduate Students

Using Head Position to Enhance Human Interfaces

by

Joel Wachman

Submitted to the Media Arts and Sciences Section, School of Architecture and Planning, on May 11, 1990 in partial fulfillment of the requirements of the degree of Master of Science at the Massachusetts Institute of Technology

Abstract

Current human-computer interface design relies too heavily on a traditional model that casts the human as an operator of a dumb machine. Methods of human-computer interaction based on the keyboard and mouse alone require reevaluation, insofar as they do not foster a personable dialogue between the user and the computer. The ideal interface uses, among other things, knowledge of the position of the user's head as a significant clue to what the computer should do next.

This thesis describes a computer program which uses a video camera to identify the position and orientation of the user's head with respect to a computer screen. Several application programs are developed to demonstrate the consequences of adding this knowledge to a traditional human interface.

Thesis Supervisor: Walter Bender
Title: Principle Research Scientist, Electronic Publishing Section

This work was supported in part by International Business Machines Corporation.

Contents

Chapter 1. Introduction

Human interfaces to computers do not embody any knowledge of the user's task. They cannot see beyond the next keystroke or click on a mouse button. Inherent in this kind of interface is a significant gap between the ideas in the mind of the user, and their implementation in the computer. The tribulations of operating a command- or window-oriented interface can create an adversarial relationship between the user and the computer, because the user must take many steps in order to communicate a trivial idea. Although some attempts have been made to show the user what the computer expects to see next, as with the Symbolics Lisp Machine's on line help, these enhancements have been shown not to increase user performance [Elkerton 1988], and they do little to foster intimate communication between the computer and the user.

In this thesis I shall describe an input device that helps the human interface act with some knowledge about the desires of the user. The device, called a *head locator* determines approximately the user's line of sight and distance from the screen, so that the interface may adjust itself to suit her. The head locator itself does not bring the human and the computer closer together. It is meant to be one component of an ideal system that does.

1.1. Those Annoying Keyboards and Mice

Keyboards and mice allow the user to convey precise information in small increments. A keyboard is used to enter one of 128 possible characters, serially. A mouse controls the instantaneous position of a cursor on the screen (despite the fact that velocity and acceler-

ation can be inferred by interpolating successive samples). The buttons on a mouse represent one bit.

The information content of any mouse motion or keystroke is very small. A single keystroke can either contribute to the construction of a word, or can be used to choose from among a number of enumerated choices. Moving the mouse makes a single significant change in the position of the cursor on the screen, regardless of the fact that the cursor must travel through all of the points between its initial position and its destination. Each keyboard or mouse event makes only a small contribution to the dialogue between the user and the computer. Each one is a necessary, but insufficient step in communicating an idea. Therefore, pressing a key, moving the mouse or clicking a button can are events that have *high precision*, but *low semantic value.*

The high precision of input devices can be a hindrance. In most window systems, for example, the user must position the cursor inside the window where she wants to direct keystrokes (the "active window"). In some systems, the mouse motion must be followed by a click on the button. SunView™ by Sun Microsystems and the Symbolics Lisp Machine™ allow the user to choose the active window by issuing a series of reserved keystrokes. This helps reduce the number of times the user must remove her hands from the keyboard, but it adds one more tedious requirement of operating the computer.

In choosing the active window, there is a vast disparity between the resolution of the cursor and the size of the object being selected. Note that at the same time, many interfaces offer painting programs which use the mouse to control simulated pens and paint brushes, or to toggle individual pixels with great accuracy.

Adjusting the mouse is a hindrance insofar as it distracts the user from her real task, which is either programming or using a computer program. Keyboards, mice, buttons (both physical and virtual), menus, and windows are distracting but necessary components of a typical human interface. Manipulating these constructs is not *part* of transferring information to the computer, in the same way as operating a manual transmission and a clutch are not *part* of travelling in an automobile. If some number of these obstacles can be eliminated, the user will be at liberty to more directly convey her intent, in the same way as an automatic transmission allows the driver to just travel.

An informal study of mouse usage among expert computer users is described in Appendix A. What can be learned from these observations is that the subjects spent , on average, about 20% of their time handling the mouse, which is time they did not spend actually getting work done. It is this unfortunate necessity of using window systems that I wish to address.

1.2. The Head Locator

The head locator provides *low precision* information by reporting the attitude of the user's head, which is considered to be of *high semantic value*. It gives the computer three important new pieces of information about the user: whether she is there all; how far away she is, and approximately where she is looking.

If a machine can infer any or all of these things by looking at the user's body, the burden of controlling it is, to some extent, taken on by the machine itself, and the user can address herself more directly to the desired task. When this knowledge is present in a computer system, the computer becomes an active participant in a dialog with the user, rather than a passive slave which must be spoon-fed instructions.

Chapter 2. Previous Work

There are alternatives to the keyboard and mouse as input devices. This chapter describes several machines allow a person to control a computer by the motion of her eyes, her hands, her head or her entire body. Of course, it is not always necessary to convey that kind of information when using a computer. Some of these devices are highly specialized and were designed with very specific applications in mind. They all claim to solve a certain kind of problem, though, which is to provide the computer with some knowledge about user's body. It is debatable whether people would want to use them habitually.

2.1. The Data Glove

The Data Glove and its sister, the Z-Glove, were developed by VPL Research to manipulate objects in computer generated virtual worlds. The Data Glove is a cotton glove fitted with up to fifteen photogoniometers that detect the amount of bend in the wearer's fingers. A Polhemus[1] magnetic positioning transmitter is fitted on top of the Data Glove, and two ultrasonic transducers are fitted on the sides of the Z-Glove. Some gloves are also fitted with piezoceramic benders under each finger which transmit a 20-40 Hz sine wave that produce a buzzing sensation at the wearer's fingertips. For a detailed description of the design of these devices, see [Zimmerman 1987]. The person wearing the glove can manipulate three dimensional images on the computer screen by using her hand. She can also receive a certain amount of tactile feedback.

[1]Manufactured by Polhemus Navigational Sciences division of the McDonnell Douglas Corporation.

The Data Glove can be used as a pointing, grasping, or gesturing device. As a pointing device, it serves the same function as the mouse, but it can be controlled in a more natural manner. As a grasping or gesturing device, the Data Glove provides a richer mode of communication between the human and computer. The data glove has been used in a variety of environments [Zimmerman 1987], [Foley 1987], [Sturman 1989], and promises to be a valuable tool for manipulating three dimensional objects in synthetic scenes.

2.2. The Graphical Marionette and the Data Suit

The Graphical Marionette Project by the MIT Architecture Machine Group [Maxwell 1983], [Ginsberg 1983], mapped the motion of a human body onto a synthetic character, the "marionette" displayed on the computer screen. The input device for this project was called the *body-tracker*. A person wore special clothing that had LEDs sewn into strategic places such as the arms, legs, and collar. The suit also had a Polhemus transmitter attached. A pair of video cameras was aimed at the user, and the output from the cameras and the Polhemus was processed by the computer to establish a correspondence between the motion of the user's limbs and the limbs of the marionette.

The Data Suit is similar to the body tracker. It uses the same photogoniometers as the Data Glove to determine the amount of bend in the wearer's limbs, and a Polhemus to determine the orientation of the wearer with respect to the computer.

When wearing either of these devices, the person *enters* the computer, in the sense that her posture, her nonverbal expression, her entire physical presence is is transmitted to the computer. Any motion she makes can be meaningful. The computer, without the addition of any artificial intelligence software, behaves that much more intelligently because

it is made privy to a large vocabulary, in fact an entire language, with which the user can communicate her intent. As [Ginsberg 1983] writes:

> The body-tracker demonstrates a system in which interaction with a computer takes place entirely within the workspace; the user is comfortably surrounded by his/her own virtual I/O device. By simply moving, one can make the computer respond. The effect is immediate and personal.

Of course, the drawback of these devices is worse than the data glove. In order to use them, the user must completely re-clothe herself.

Neither the data glove nor the data suit are useful input devices for routine programming tasks, because they require the user to make special preparations to use the computer. They are designed for applications in which the user must manipulate objects in a virtual world, or provide gestural input to the computer.

2.3. The Eye Tracker

Eye trackers are a third kind of input device that attempts to provide a connection between the user's body and the computer. The intent of an eye tracker is to discover exactly where the user is looking. There are four common ways of tracking the motion of a person's eyes. One method is to place electrodes on her face to measure changes in the orientation of the difference in the electrical potential between the cornea and retina. This method only gives a reading when the eye moves, and it is not very accurate.

Another invasive method is to place contact lenses on the subject's eyeballs. The lens has a small mechanical switch, lever, coil or mirror attached to it so that it can be tracked. According to [Jacob 1989], it is "very awkward and uncomfortable, covers only a limited range, and interferes with blinking." Unfortunately, this is the most accurate method.

The least uncomfortable methods are single- and two- point optical/video trackers. These methods use some remote sensing device to locate some feature of the eye, such as the boundary between the sclera and iris, the outline of the pupil, or the reflection of light shone directly onto the cornea. Single point measurement tracks only one of these features, while two point methods track two features, which allows the system to distinguish between eye and head movements. Single-point tracking methods require that the head remain perfectly still, while two-point tracking methods allow the subject a certain amount of freedom of head motion.

Jacob describes one commercially available eye tracker:

> The most reasonable method is the corneal reflection-plus-pupil outline approach, since nothing contacts the subject and the device permits his or her head to remain unclamped. In fact the eye tracker sits several feet away from the subject. Head motion is restricted only to the extent necessary to keep the pupil ...within view of the tracking camera. The camera is panned and focussed by a servomechanism that attempts to follow the eye as the subject's head moves. The result is that the subject can move within approximately one cubic foot of space without losing contact with the eye tracker. ...Because [infrared light is used], it is barely visible to the subject. The video image of the pupil is then analyzed by analog image processing circuitry, which searches for a large, bright circle (pupil) and a still brighter dot (corneal reflection). Then, a computer calculates the center of each and, from the two points, determines the line of gaze.

The most difficult problem with an eye tracker is that it cannot accurately pinpoint the user's center of attention. Most eye trackers have filters built in to remove the microsaccades and high frequency tremors that naturally occur when the eye is focused on one spot. However, a person's eye never fixes on a single point for very long. Her attention constantly shifts about the screen (see Figure 2.1). Jacob calls this the "Midas Touch" problem, because the computer interprets every eye movement as meaningful, so the user

must be extremely careful not to look at anything she does not want to affect. The Midas Touch remains an unsolved problem with eye tracking.

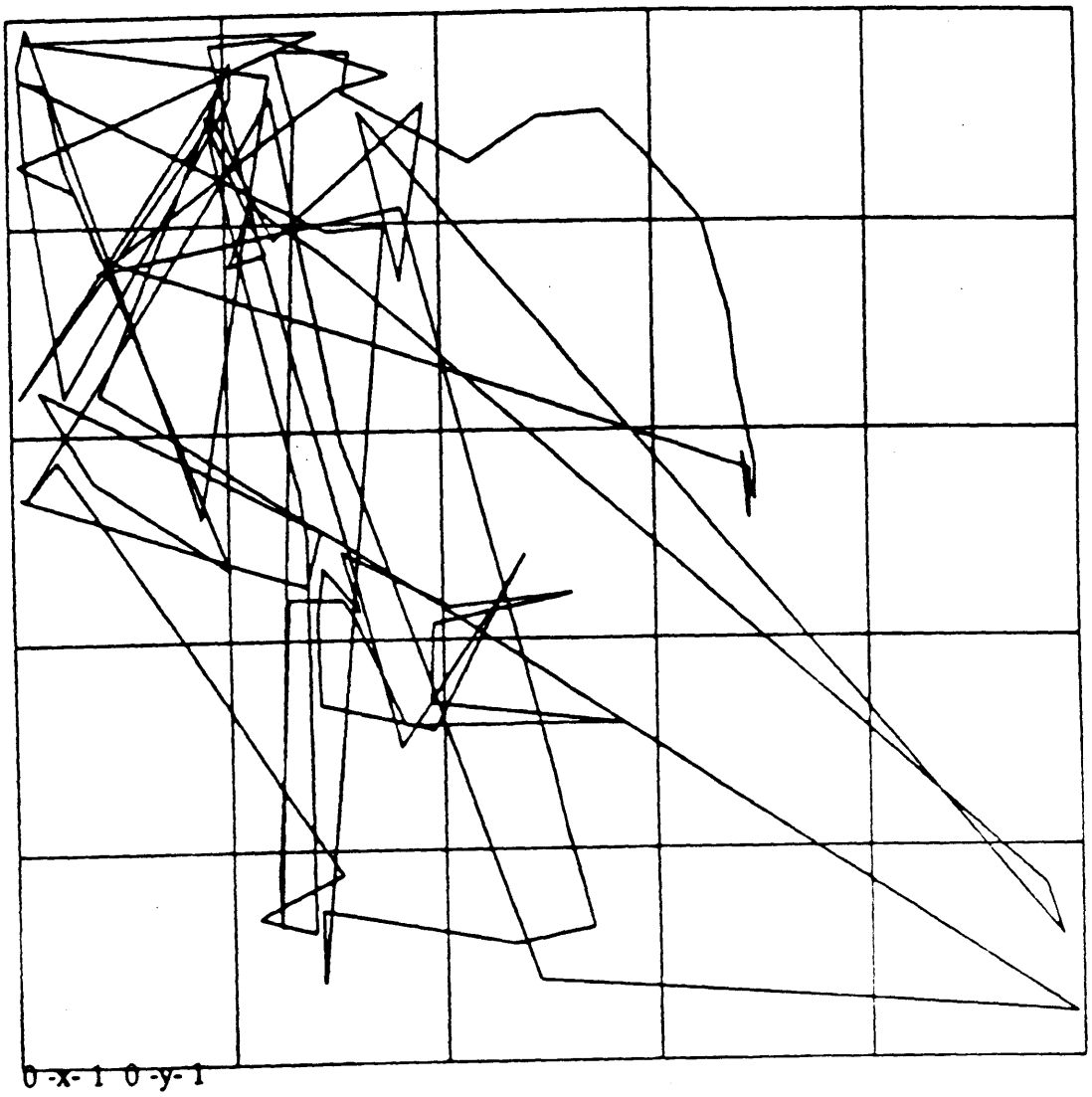


Figure 2.1 Eye Motion Recorded by an Eye Tracker (From [Jacob 1990]).

2.3. Head Mounted Displays

As early as 1968, Ivan Sutherland [Sutherland 1968] built a head mounted display for viewing 3-dimensional objects. The display itself consisted of a pair of CRTs mounted in front of the user's eyes. Computer generated stereoscopic wireframe objects were shown in the CRTs, and the user's view of these objects changed as she moved her head. Sutherland used two methods to locate the head. The mechanical method:

> ...involves a mechanical arm hanging from the ceiling...This arm is free to rotate about a vertical pivot in its ceiling mount. It has two universal joints, one at the top and one at the bottom, and a sliding center section to provide the six motions required to measure both translation and rotation. The position of each joint is measured and presented to the computer by a digital shaft position encoder.

However, it was "rather heavy and uncomfortable to use". Sutherland also built an ultrasonic head position sensor which resembles the modern Polhemus sensor, in theory. This device allowed the user unencumbered motion within a 6' x 3' volume.

Despite its drawbacks, Sutherland's system worked well. He remarks:

> Even with this relatively crude system, the three dimensional illusion was real. Users naturally moved to positions appropriate for the particular views they desired. For instance, the "size" of a displayed cube could be measured by noting how far the observer must move to line himself up with the left face or the right face of the cube.

In another demonstration a large wireframe room was displayed. Users could step "inside" the room and look around, and the view would change in the expected manner. One drawback of the system was that it displayed everything in wireframe, so some objects

Photos of Ivan Sutherland using his head mounted display

FIGURE 4—The ultrasonic head position sensor in use

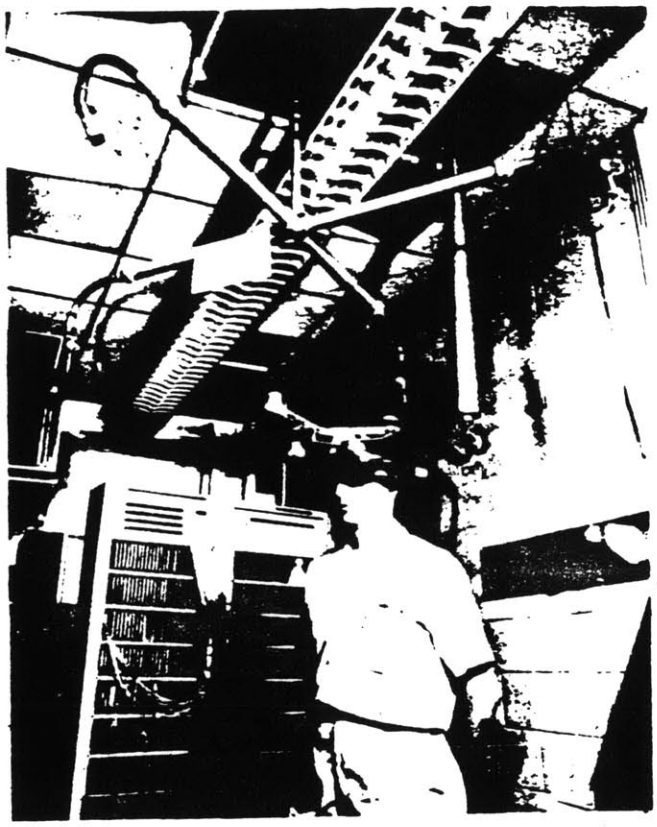
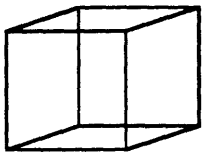


FIGURE 3—The mechanical head position sensor in use

would appear ambiguous. For example, every cube showed the Nekker phenomenon, in which it is impossible to tell near faces from distant ones.[2]



A Nekker Cube

The modern incarnation of the head mounted display is built by NASA. It is helmet similar to a motorcycle helmet, that is equipped with a Polhemus and an LCD display.

As a Master's thesis for the Architecture Machine Group at MIT, Scott Fisher [Fisher 1981] built a Viewpoint Dependent Imaging Display, which he described as:

> ...a viewpoint dependent imaging system in which a video monitor or projection screen becomes a virtual window into a three dimensional viewing environment...The resulting display is continuously updated to present a perspective corrected, lifesize, 3D image that is under user-control.

Fisher's device consists of two videodisk players, a computer, and a Polhemus. The videodisk contains several thousand stills of the same scene, shot from slightly different viewpoints. The position of the user, relative to the display, is measured by the computer and Polhemus. The still taken from the corresponding camera position is displayed on the screen.

The machine accounts for left and right, forward and backward motion of the user's head, relative to the display medium. It does not change the view when the user tilts or nods her

[2]The solution to this is what [Hagan 1968] called "Intensity Windowing". On his Adage Graphics Terminal, a vectorizing display, the intensity of each vector drawn is proportional to its apparent distance from the user.

head. Despite these limitations, the it is an important early example of the relevance of the user's viewpoint in computer controlled environments.

In the years since [Sutherland 1968], head mounted displays have been vastly improved. However, all of the tools to help people wander through artificial reality must be worn or carried, and they are often cumbersome. Despite the fact that the head locator was not developed with artificial reality in mind, I hope that the spirit of this research is carried into that field.

All of the devices described so far are clever and can even be useful in some circumstances. However, they have a common drawback: they are not at all convenient to use. In some cases they are downright impossible. The problem with all of them is that not enough thought was given to how they would be used outside of the laboratory. Too often, a new technology is developed for the sake of performing some previously unavailable function, and little or no thought is given to its eventual use. In some cases this brings up thorny moral or philosophical questions regarding its possible applications. In other cases the invention is innocuous enough, but it is too unwieldy or complicated to use. When developing a new technology it is important to consider the ramifications, not only of its applications, but of what it will require of its users.

2.4. Elegant Devices in the Mundane World

Devices that respond to human body positions do not need to be as elaborate or arcane as eye trackers and head mounted displays. Several very simple gadgets that have nothing to do with computers have been developed for use in contexts where it makes sense to know where a person is.

Olivetti Research designed an "active badge" that locates individuals within a building. The badge helps the PBX (Private Branch Exchange) to forward an incoming telephone call to the telephone nearest the person for which it is intended. In effect, this associates each person with a number that will always reach her, rather than the telephone in his office. The active badge transmits a unique signal for each person to multiple receivers scattered throughout the building. The system that monitors the signal can determine whether a person is present in the building, where she is and, possibly, what she is doing, at any time while the badge is worn and powered on.

In the men's room at O'Hare International Airport in Chicago, a photosensor is mounted in the wall above each urinal. When a person enters and then leaves the vicinity of the photosensor, the urinal is automatically flushed. This very simple presence detector performs a necessary, repetitive task which benefits the user, who does not have to get his hands wet or dirty by touching a public valve, and benefits the airport by ensuring that the urinals are always flushed, and by requiring less maintenance. I will grant that attaching a presence detector to plumbing seems frivolous, but this application illustrates the fact that watching people is an idea that has convenient and logical ramifications in everyday life.

These two are simple and non-invasive devices, and they can largely be ignored by the user while they are operating. These qualities make them very attractive components of a human interface because they were designed with their applications in mind.

Chapter 3. Aspects of Human-Computer Interaction

The interface between the user and the computer may be the last frontier in computer design.

- James D. Foley

In this chapter I will examine some models of human-computer interaction with the intent of developing some ideas about the ideal human interface. I will begin by exploring some methods of evaluating the quality of an interface in terms of human performance models. I will show how previous models of human-computer interaction have hindered understanding between the user and the computer. Finally, I will examine a theory of "intelligent interfaces", as described by [Chignell 1988].

3.1. Historical Perspective

In the early days of computing there was very little contact between the user and the computer. In fact, scientists and engineers who needed to perform some calculation had to participate in a rather complicated dance in order to get results. The scientist formulated the calculation in mathematical terms, and then gave those pages to a programmer. The programmer translated the mathematics into a computer language (usually FORTRAN) and handed the program to a keypunch operator. The keypunch operator translated the computer program into stacks of cards off line, and handed the completed stacks to a computer operator, who (finally) fed them into the computer. Some time later, the results appeared on sheets of perforated paper delivered to a central output window, where they were collected by a research assistant and delivered back to the scientist (user). If an error

occurred in any one of these steps, the entire procedure would be repeated from the beginning.

This arrangement did not foster fluent communication between the user and the computer. Of course, the limited computing power of early computers did not invite such a relationship, even when time sharing was first introduced. The increase in computing power and the widespread availability of time sharing terminals brought the computer and the user closer together, and with this proximity a new kind of relationship began to form. It was at this time that the psychology of human-computer interaction first became an issue in computing.

Bæker and Buxton [Bæker 1987] note that the term "software psychology" was coined in the late 1960's to describe:

> those aspects of human-computer interaction that are particularly concerned with the way humans interact with software structures and with the effect different structures and systems have on human behavior.

In 1974, the *Applied Information-Processing Psychology Project* was formed at Xerox PARC, the aim of which was to develop an empirical formulation of the psychology of human-computer interaction. The results of the group's work were published in [Card 1983], and represent the first scientific study of the way that people use computers. The timing of this work is significant, because it was published at the period between the phasing out of line-oriented hardcopy teletypewriters, and the widespread introduction of window systems[3]. It is at this point that the relationship between the human and the com-

[3]The first window system, the Xerox STAR, was developed in 1981. Window systems did not become widely available as a user interface until the introduction of the Apple Lisa™ and Macintosh™ in 1984.

As a measure of where [Card 1983] fits in the history of computing, it is interesting to note that they write,

puter changed from that between an operator and a machine, to something quite different and theretofore unknown. Card, Moran, and Newell were among the first to describe this new relationship:

> The key notion, perhaps, is that the user and the computer engage in a communicative dialogue whose purpose is the accomplishment of some task. It can be termed a dialogue because both the computer and the user have access to the stream of symbols flowing back and forth to accomplish the communication; each can interrupt, query and correct the communication at various points in the process.

This new form of communication was alien to the entire computer community. A few pages later, they comment,

> First, interaction with computers is just emerging as a human activity. Prior styles of interaction between people and machines...are all extremely lean: there is a limited range of tasks to be accomplished and a narrow range of means...But the user... does not operate the computer, she communicates with it to accomplish a task...What the nature of this arena is like we hardly yet know. We must expect the first systems that explore the arena to be fragmentary and uneven.

The first attempts at building human interfaces were uneven, indeed, insofar as they continued to operate under the operator/tool paradigm. The designers did not build interfaces that cultivated predictable, consistent communication with the computer. There are many examples of the way poorly designed interfaces can pave the way to disaster. In one system the command "EDIT" caused the computer to select Everything, Delete the se-

---

> Immense gains will occur when the display holds not the common 24x80 characters (the typical alphanumeric video terminal, widely available today), but a full page of 60x120 characters (the typical 1000x800 pixel video terminal, available at a few places today), or even the full drafting board of 512x512 characters (not really available anywhere, yet, as far as we know.)

Although no screen known to the author can display 512x512 characters of 8x8 pixels apiece, great strides in display technology have been made in the seven years since [Card 1983], so that it is now possible to show up to 250x250 characters on a commercially available screen.

lection, and Insert the letter 'T'. There is nothing intuitive or natural about such an interface. These kinds of design foul-ups prompted [Bæker 1987] to comment:

> Today's interfaces are not even half-witted. They are downright intolerant, insensitive and stupid.

It is telling that Bæker describes the computer's behavior in human terms. Once the channel of communication between humans and computers was opened, the expectation was that computers would speak a human language. In reality, the result was a sort of culture clash which has not yet been fully resolved.

3.2. Measuring Performance

The work begun by Card, Moran and Newell provided a basis for the evaluation of user performance. The authors considered the human being to be an information processor that can be measured in similar ways to a machine. They acquired experimental data for the basic relevant components of the human information processor: perceptual response times (minimum frame rates, morse code listening rate, reading rate, etc.); motor skill (Fitt's Law[4], Power Law of Practice[5], keying rate, etc.); and cognitive acuity (memory rates, decision rates, pattern matching rates, etc.). The result was an objective basis upon which to formulate a theory of human performance, given a known interface.

According to performance theory, the goal of the human information processor is to accomplish some task. This process can be broken down into four subtasks: *Goals*, *Operators*, *Methods* and *Selection-rules* (GOMS).

---

[4]Fitt's Law is a measurement of the amount of time a person needs to move his hand from one place to another. It takes into consideration the amount of time needed by the Perceptual, Motor and Cognitive subsystems in the human information processor model.

[5]The Power Law of Practice states that the time to do a task decreases with practice.

Under the GOMS model, the user has a specific goal in mind which is to complete a particular task such as finding a document in a document retrieval system. The user breaks down her goal into smaller subtasks (Operators) such as organizing her thoughts, selecting a search domain, and figuring out how to use the computer. The user then selects some Methods by which to accomplish each subtask, such as deciding which program to use and which subjects to search. The Methods are chosen by Selection-rules which the user acquires through experience or by rote.

The Goals and Operators are largely independent of the computer system being used. The role of the human interface is important in determining the Methods and Selection-rules. With experience, the user establishes better methods because her understanding of the functioning of the computer becomes gradually clearer.

By using the GOMS model of performance in combination with the human information processor model of human behavior, it is possible to make an accurate estimate of the amount of time it should take to accomplish any Goal, given a set of Methods. The amount of time required by any particular Method is measured by using the Keystroke Model or the Unit-Task Analysis Model. The Keystroke Model measures the total amount of time it would take to enter the commands necessary to accomplish a Subgoal using a particular set of Methods. For example, the Keystroke Model measures the amount of time it takes to key in a computer program using the Unix editor VI versus EMACS, each of which is controlled by an unique set of keystrokes. The Unit-Task Analysis is an approximation of the same time requirement, based on examples of the times required to accomplish the component subtasks in similar systems. For example, the time necessary to lay out a page using a graphical interface can be measured in terms of the average time necessary to choose fonts, enter text, place headlines, and so on, measured over several well known systems.It is therefore possible to determine which

Methods will result in higher performance. Furthermore, the construction of the interface affects performance insofar as it encourages some Methods over others.

3.3. Knowledge Representations and Mental Models

People strive to understand computers by establishing analogies. The application of proper analogies will generally help a person to implement the proper Selection-rules for determining what to do next. In their examination of mental models of human-computer interaction, Carroll and Olson [Carroll 1988] show that an appropriate mental model increases performance.

Studies of human-computer interaction usually begin with an examination of the kind of knowledge the user has about the computer. Carroll and Olson propose three levels of knowledge representation. On the simplest level, people use *rote sequences* to communicate with the computer. They require specific instructions, like a recipe, and have little or no knowledge of the effect their instructions have on the internal organization of the software.

More experienced users may have some preferences about what *methods* are best suited to a task. Knowledge of sequences is implicit in knowledge of methods. However, methods may also be applied by rote, still without any understanding of the computer's internal organization.

Finally, a user may develop (or be given) a *mental model* of the internal structure of the computer. A mental model is an internal representation of what the user thinks is happening inside the computer. Carroll and Olson describe four kinds of mental models, *Surrogates, Metaphors, Glass Boxes and Network Representations.*

A Surrogate is a conceptual object that mimics the system's behavior, but does not assume the way in which the output is produced in the surrogate is by the same process as in the system. It is an analogy that allows the user to predict the system's reaction to certain stimuli, but does not give the user any useful knowledge about the structure of the actual system. A block diagram consisting of the standard symbols for boolean logic is a surrogate for a logic circuit. The diagram shows what the circuit does, but not how it works.

A Metaphor is a direct comparison between the system and something else in the world with which the user is already familiar. One common metaphor is, "a text editor is a typewriter". A metaphor gives the naive user a familiar language with which to understand the computer system. It does not need to be accurate, but an inadequate metaphor will detract from the user's understanding of the computer's behavior, and reduce performance.

A Glass Box is a construct that is somewhere between a surrogate and a metaphor. It is like a surrogate in that it mimics the system perfectly, but it is like a metaphor insofar as it offers new labels for the components of the system. According to Carroll and Olson, glass boxes have mostly been used in *prescriptive* contexts (eg., a user is instructed to think of input as a ticket window), and are not habitually generated by users independently (ie., they are not *descriptive.*)

Network Representations are connected graphs with nodes that describe the possible states of the system. The lines between the nodes indicate the kinds of actions that can be taken to move from one state to another. The network representation described in [Carroll 1988] is a *Generalized Transition Network* (GTN). A GTN is like a Surrogate, because it shows exactly the action/reaction states of the computer, but it does not explain why the system is set up that way. By examining the user's own internal GTN, and matching it with the real GTN, one can discover areas where the user is having trouble understanding the structure of the system.

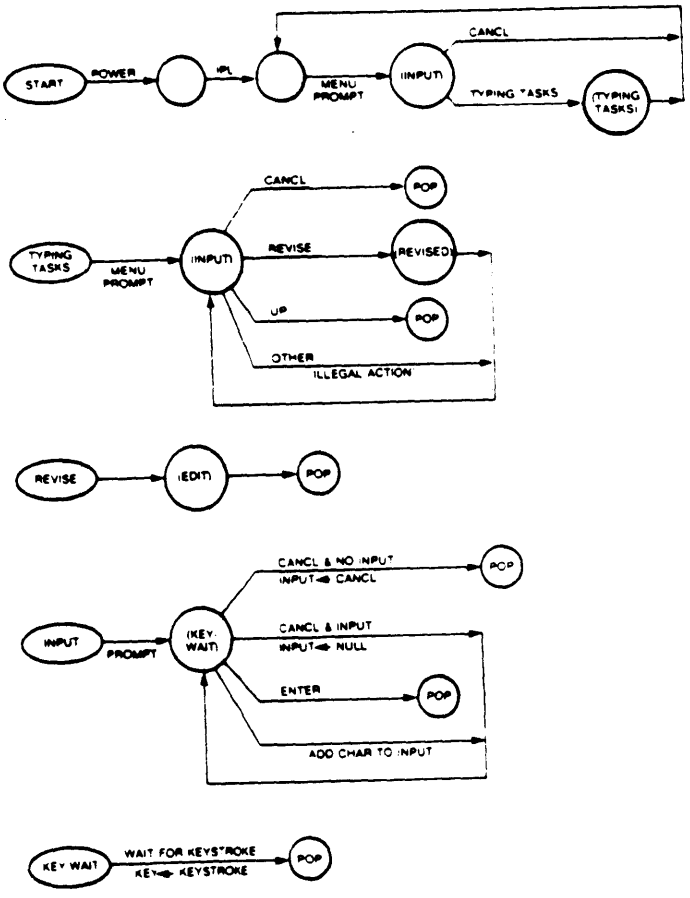


Figure 3.1. Example of a Generalized Transition Network (GTN) [Carroll 1988].

3.4. Intelligent Interfaces

The problem with concentrating on understanding the user's mental model of the computer system is that it does not in any empirical way indicate how an interface should be designed. Understanding users' concepts of existing interfaces is a descriptive, not a prescriptive practice. The whole development process should be turned around. Instead of making the user's mental model conform to the computer, the computer should be given a mental model of its users.

The Intelligent Interface is the embodiment of this philosophy. Chignell and Hancock [Chignell 1988] define an intelligent interface as:

> ...an intelligent entity mediating between two or more interacting agents who possess an incomplete understanding of eachother's knowledge and/or form of communication.

An intelligent interface contains a mental model of the user, as well as a model of the computer, and acts as a translator between them. The interface can be a human being or it can be software. An example of an intelligent human intermediary is a reference librarian with access to a computer document retrieval system. In order to locate a specific document, the unskilled user explains to the reference librarian what kinds of documents she needs. The reference librarian translates the user's request into something the computer can understand, and narrows the search by continually switching between the user and the computer until the documents are found.

This intelligent intermediary must have several kinds of knowledge in order to "translate the intentions of the user faithfully and to enhance the user's understanding of the system

output". The most relevant of these that are classified by [Rissland 1987] are knowledge of the user and knowledge of the user's tasks. Knowledge of the user consists of knowledge of her expertise, style, preferences and history. Rissland illustrates that a novice programmer needs different support than an experienced one. Knowledge of the user's tasks consists of knowledge of the context, purpose and use of the result of the task, such as whether the user is a manager or a secretary.

Rissland warns against working towards an interface that tries to do too much. She writes:

> Yes, the interface must have access to many sources of knowledge...but it is not useful to load all the problems of intelligence into the interface, as if the interface contained a "hidden homunculus". Where the critical boundary is for off-loading too little or too much intelligence onto the interface is not settled: in fact, it probably changes in response to many factors.

Expert systems, natural language interfaces and hypermedia are attempts at quasi-intelligent interfaces. The first two certainly do embody an amount of knowledge about how people communicate. However, they do not contain models, *per se*, of the interconnectedness of human thought. This is best illustrated by the inability of most natural language processors to resolve anaphora, or to make inferences based on context.

Vannevar Bush [Bush 1945] described a system for document retrieval that would link documents in an associative manner so that they are connected to other documents that share concepts. Bush's "Memex" was the first description of what has come to be known as a hypertext system. According to Cignell, this form of storage seems to conform with the way human memory is organized. Whether or not this is true, it has not typically been the case that hypertext systems organize knowledge in the way we are accustomed to thinking. Most users find themselves easily lost in these kinds of interfaces, because they need to maintain mental map of the internal graph structure. Chignell and Hancock disagree, making a surprising statement about human-computer interaction:

> Regardless of how the conceptual representation and intelligent interface is defined, there will almost always be some discrepancy between the user's model of the task and the model actually reflected in the interface. In many cases it would be inappropriate to attempt to incorporate an unmodified user model in the interface, since this model my be faulty in some respects. Thus there is a need for training to modify the user's model so that it conforms more closely to the accurate model of the task as represented in the interface.

They go on to propose that hypermedia can provide the necessary "training" by forcing the user to adopt the computer's model. This attitude is downright frightening. It conjures up Orwellian visions of the world in which people's mental models of the world are "modified" to suit their computer masters. To a degree, it is easy to force a user to think in a particular way about an interface. My question is, what does this accomplish? If the human and the computer are not in sympathy, their communion is cold and remote, and little work will get done at all.

Human interfaces should include internal models of their users, not the other way around. If this were the case, each person would have the flexibility to communicate with the computer in her own private manner. The truly intelligent interface would harbor expectations about the user based on her identity, in the same way as each person has expectations about the people she knows. No interface yet developed has given us the ability, as [Licklider 1960] writes, "to think in interaction with a computer in the same way that you think with a colleague whose competence supplements your own...".

Chapter 4. An Alternate Design Philosophy

If an intelligent interface is to be of any use it must act like a surgeon's assistant. It must be an unobtrusive but indispensable partner that helps the user perform her task without getting in the way. The conventional human interface paradigm is not conducive to developing such systems. Most human interfaces require constant manipulation of the keyboard and mouse. The nature of this type of interaction was discussed briefly in Chapter 2, in which I concluded that human interaction using these devices is inadequate owing to the low semantic value of any single event. In this chapter I explain how standard interfaces make use of the phenomenon of selective attention, and how two concepts in particular that emerge from this approach necessarily limit the depth of human/computer interaction. I conclude by describing an alternate design philosophy that should guide the development of more autonomous interfaces.

4.1. Selective Attention and the Overmanaged Interface

An assumption that underlies the design of most computer interfaces is that a person can only attend to one thing at a time. This claim has been made by many researchers concerned with the psychology of human-computer interaction, including [Hochberg 1970], and [Wærn 1989]. As a result, conventional interfaces use a single cursor and allow input into a single window, and assume that the user will only be interested in one spot on the screen at any time. Arguments supporting the claim are compelling, and have to do with the nature of selective attention, which we shall examine further.

4.1.1. The Phenomenology of Attention

Hochberg describes attention as *selective* and *focused*, and having an element of *anticipation*. Selectivity is the quality of perception that allows us to report only a subset of the data available to our senses. Attention is focused in the sense that one can attend to only one thing at a time. He calls this the "most striking attribute of attention", and Wærn confirms it with a trivial observation:

> The question of selecting a particular stimulus is natural to our visual system, since the eye cannot focus on more than one spot at a time.[6]

It is true that the eye only focuses on a single spot, centered in the fovea. However, it is possible to perceive stimuli both within the fovea and in the periphery. The information is present, but whether or not to ignore it is a matter of choice.

Anticipation is described as that quality of attention that establishes a context in which a sensation will be perceived. It is a kind of process of initialization, or setting up filters to make the perceptual apparatus ready to accept a certain kind of input. It is an attribute of very low levels of perception, part of the perceptual apparatus, rather than a con-

---

[6]Wærn goes on to say,

> It is equally natural to the motor system, which is unable to perform more than one single muscle action at the same time.

Certainly no dancer would agree. She continues,

> The auditory system, on the other hand, allows for attention to two different stimuli simultaneously. We can therefore ask ourselves whether selective attention also applies to auditory stimuli.

This statement also rings false, and Wærn later contradicts herself by citing an experiment in which a person presented with two simultaneous auditory stimuli is unable to report more than one of them accurately.

scious cognitive task. When listening to someone speak, for example, the auditory system predicts which phonemes are about to be spoken, based on the current context. Anticipation is related to the Gestalt theory of perception, which explains that the visual system tends to see things in groups, and tends to have certain expectations about the relationships among objects in a group. Optical illusions are sometimes the result of fooling the system into anticipating certain things. Figure 4.1 shows three shapes that cause the visual system to anticipate a fourth shape. The anticipation has been induced by the arrangements of the shapes, but not met in reality.

I will grant that it is useful to exploit this property under some circumstances. The connection between the mouse and the cursor appeals to this description of attention because the user accepts the events on the screen into her own sense of reality. The mouse and the cursor form a single object, as a collection of phonemes forms a word. The coupled mouse and cursor are considered to be a particular kind of *thing*, the nature of this *thing* being contingent upon the isotopic motion of its parts. A different kind of behavior could only arise out of the actions of a completely different *thing*. That is, if the cursor begins to move non-isotopically with the mouse, or if it moves without the user's consent, what is broken is not only a trust, but a sense of contingent reality.

4.1.2. Attention and the Interface

The claim that attention is always selective and focused should not go unchallenged. Selective attention is only one of many levels of human cognition. A person invokes selective attention to choose from among a large number of stimuli, such as listening to one particular conversation at a crowded party. According to Hochberg, a person can decode only one conversation at a time. Introspection shows us, though, that it is possible to simultaneously listen to the sound of the crowd, respond to a question, taste an *hors*

d'oeuvre, and flirt with an attractive person across the room. It is a matter of choosing the level of granularity. In human interface design, the conventional grain is too fine. Human interfaces should be designed to take advantage of the fact that human beings can in fact control several input devices and comprehend several modalities of output concurrently.

In a rich computing environment, one in which more than one program is running, the user's attention will wander about the screen, and even outside of the computer workspace. Hochberg and Wærn would claim that, regardless of this fact, the user is only capable of attending to one event at a time. They would argue that while the user clicks on the mouse in order to, say, choose an item from a menu, she is incapable of performing any other task. This is simply not the case. It is the design of conventional interfaces that is responsible for that limitation. It is possible, for example, for a person to control the mouse and be aware of the arrival of electronic mail, as indicated by the change in state of an icon which she can see in her peripheral vision. If the system were equipped with the proper tools, she could continue to control the mouse and reach out to press on the mail icon in order to confirm that she has received the message. The sensation would be similar to other cases in which a person does two things at once, such as reading a book and drinking a cup of coffee.

User interfaces that force the user to into selective attention force the human/computer interaction to be serial. The natural parallelism of human performance is lost. This can be thought of as a loss of communication bandwidth.

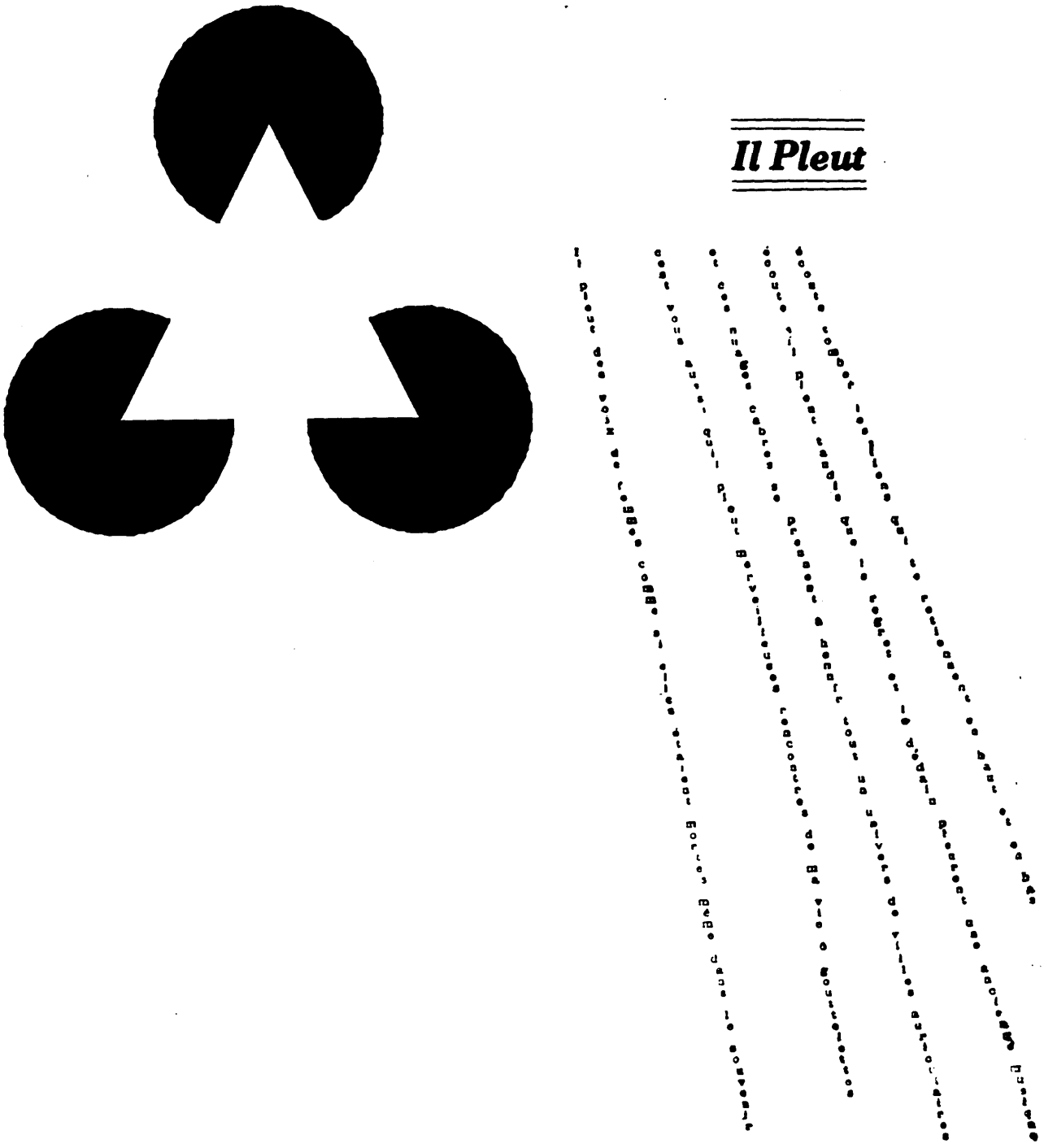


Figure 4.1.
(a) An optical illusion that illustrates perceptual anticipation. The visual system invents a shape that is not really there.

(b) The poem, "Il Pleut" (It is Raining) by Apollonaire exploits the reader's ability to select a level of attention. The arrangement of the words suggests falling rain. This poem is an example of a genre of poetry called "calligrams".

4.2. Point of Control and Peripheral Events

Two concepts are endemic to serial user interfaces. I call them the Point of Control, and Peripheral Events. The Point of Control (POC) is the place on the screen where commands to take effect. The current cursor location is the Point of Control when one manipulates the mouse. In a window system, the active window is the point of control.

Peripheral Events are events that occur somewhere other than at the POC, and are independent of the user's intentions at any moment. An example of a peripheral event is the icon that changes its appearance when electronic mail arrives. The change of state of this icon is a peripheral event because it is not initiated by any direct action by the user, and it is not likely to occur near the center of the user's attention.

4.2.1. Point of Control

The Point of Control (POC) corresponds to the user's center of attention and shifts unpredictably, as far as the computer is concerned. There is no practical way for a computer system to predict with any certainty where the user is going to look next, or with which of the active programs she will interact. The Apple Macintosh™ operating system (as well as other systems which it subsequently inspired) attempts to alleviate this problem by restricting the user's attention to one program by presenting a "modal" dialogue, which insists that input be directed to that program before she can do anything else. Modal dialogues are a throwback to command driven interfaces in which commands must be entered serially. They defeat the purpose of a multitasking graphical interface.

The fault lies with the computer, not with the user. Systems do not have to get very complex before it becomes practically impossible to predict what the user is going to do next. This is why the user must operate the computer by an interminable series of keystrokes and mouse clicks.

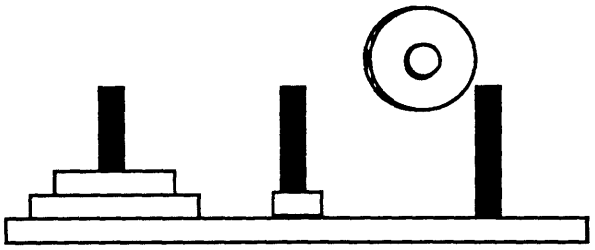
4.2.2. Peripheral Events

A Peripheral Event is any event that is not initiated by the user and is not the direct result of an action taken by the user at the Point of Control. On the simplest level, this means that the twinkling and beeping of background processes unrelated to whatever the user is doing at the moment are all Peripheral Events. An event need not be spatially separated from the POC to be peripheral. The salient feature of a peripheral event is that it happens irrespective of the input/output exchange at the POC.

The quality of Peripheral Events ranges from the urgent to the mundane. An example of an urgent event is the notification of a pending system shutdown. In current systems, it might be indicated by a message in an output-only window, or by an audible "beep". The change in state of the Mail icon mentioned above is an example of a mundane peripheral event.

Peripheral Events are most often events that are designed to call the user's attention away from the Point of Control, or to offer the user the opportunity to move the POC elsewhere. It is acceptable for an interface to call the user's attention to something happening in the background, or elsewhere on the screen. Unfortunately, given the way most computer systems are designed, the user must change modalities in order to move the point of control. She must move the cursor to a particular place on the screen, or deactivate a window, or activate a menu, or she must do several of these things, and in the process solve a

kind of Tower of Hanoi problem in order to juggle the interface to get the desired effect. She must solve questions like, "where should this window go if I don't want that one to be covered?", and "which menu contains just the command I'm looking for?". This sort of thing gets in the way of the task.



The Tower of Hanoi puzzle in progress

4.2.3. Eloquent Scenery

In his Master's thesis [Mollitor 1990], Robert Mollitor explores a particular subclass of peripheral events that are designed to communicate to the user in the peripheral field of vision. He calls this subclass "Eloquent Scenery". These kinds of peripheral events are a bit of a compromise, insofar as they do not necessarily demand the user's full attention, for they never call for a change in the Point of Control. Eloquent Scenery consists of various moving icons called "Gestures" that express messages of various levels of urgency, depending on their shape, color, content and path of motion and the way these qualities change in time.

The impact of Gestures has a great deal to do with their placement with respect to the user's center of attention. Mollitor admits that development of that aspect of Gestures was somewhat hindered because the computer did not know much about the user's location:

> The system, currently lacking a way to detect the user, assumes the presence of the user. More sophisticated systems could both detect the presence and activity of the...user and track a specific user throughout the extent of a communications network.

If the human interface knows where the user is looking at any moment, it can place the Gestures in meaningful positions, depending on their urgency. The most urgent messages could appear directly in the user's foveal field, whereas less important ones could be placed further in the periphery.

Gestures can also be status indicators for things like current system load or the amount of completion of a background compilation. If they are used in these ways, Gestures transcend the traditional Point of Control/Peripheral Event model, because they assume nothing about what the user is going to do next, and they do not need to be controlled, yet they convey useful information.

4.3. The Autonomous Interface

We should question conventions when they outlive their usefulness. The model of human/computer interaction that gives rise to the Point of Control is no longer useful because it exploits too little of the human ability to control more than one thing at a time. In addition, the fact that Peripheral Events exist is an indication that the computer cannot, in many circumstances, continue to operate without the user telling it what to do. Peripheral Events are like the cry of a small child who needs attention.

An intelligent interface should have a certain amount of inertia, so that it can be shoved in a particular direction and keep going indefinitely without further intervention. This might be called an "autonomous" interface, because it does not constantly depend on the user. The autonomous interface should:

- accept multiple inputs simultaneously;
- accept input/suggestions from the user at all times, anywhere on the screen;
- figure out what the user is trying to do, and assume that she is going to continue doing it;
- present important information where the user is looking;
- be an intelligent interface.

Although I am not aware of any true autonomous interfaces, interactive newspaper projects developed by [Salomon 1983], [Donath 1983] tended towards that goal. Donath's EMAG program eliminated the cursor by accepting gestural input through a touch sensitive display. All of these programs provided the user with information the *computer* thought was relevant.

The head locator was built with the autonomous interface in mind. Autonomous interfaces must incorporate some knowledge about the presence and attitude of the user in order to predict what the user is going to do next, and to locate information based on where the user is looking. Head locator applications were built to serve as examples of the properties of an autonomous interface.

Chapter 5. The Head-Server Vision and Communications Subsystems

Systems that attempt to glean some information about the orientation of the user have been developed by [Sutherland 1968], [Rabb 1979], [Bolt 1980], [Jacob 1989], [Mase 1990], and others. The most common way of detecting head or hand motion is to mark the user in some way which makes it easier for the computer to see her. This has been done by placing a Polhemus on the user's head or hand [Bolt 1980], or by marking the user's face with brightly colored crosses [Rabb 1979] and then shining bright lights at her face. It could also be done by requiring the user to wear a striped mask, or to shine an infrared light into her eyes. All of these methods are invasive, if not downright unhealthy, and they hinder the user to such an extent as to defeat their own purpose. A reasonable goal, therefore, was to build a system that would work under average room lighting, that would not require the user to wear any special clothing, or to be marked or physically constrained.

5.1. Design Philosophy

In designing a platform with which to demonstrate the head locator I decided to use an existing interface, X Windows, because it is a well known interface and I had already had some experience working with its source code. The fact that X Windows is well known is a boon, because it is important to demonstrate that a head locator can dramatically enhance the ability of a common interface to communicate with the user. With the addition of knowledge about the user's position, the computer begins to make choices based on

what it believes is best for the user, which is the kind of behavior expected in an intelligent interface.

In related work, [Schmandt 1989] developed a voice interface to the X Windows system which allows a user to choose which window becomes active by saying its name. This is a significant step for a few reasons. Primarily, it means that the computer can recognize the user's voice, so it has some idea about who is using the interface. It means that the user has another way of communicating with the computer. Lastly, according to Schmandt, the addition of a voice interface "allowed users a greater range of physical motions and positions, since they were not so tied to the keyboard and mouse." Several people have reported that the voice interface was difficult or confusing to use.

It would have been desirable to build an autonomous interface to demonstrate the head locator. However, the design of such an interface is a large, complicated project that is beyond the scope of this thesis. As a compromise, a spectacular but limited demonstration could have been accomplished by building a specialized X Windows window manager that used the head locator instead of a mouse. In the end, however, it was more logical to add the head locator to a preexisting interface in order to show how much a conventional interface can be improved by the simple addition of knowledge about the user.

The head locator was designed as a kernel of software that runs in the background, as a server. The ***head server*** software consists of a *communications subsystem* and a *vision subsystem*. The communication subsystem is responsible for accepting and responding to requests from client programs that need information about the user's head. The vision subsystem is responsible for peering through the video camera and locating the head. The user's head is viewed in profile, against a solid background. A description of the analysis performed by the vision subsystem follows in section 5.4.

5.2. Observations of Head Gestures

In order to determine a reasonable rate at which the head server should run, I chose to determine roughly the rate at which people gesture with their heads. The aim was to make the system fast enough to detect motion just below the rate of head gestures, which would be the fastest people would move their heads to seriously look at something. It was also desirable to filter out motion that is faster than head gestures, so that the system would not be confused by quick glances. It was not my intention to build a system that was too slow to detect head gestures. Neither was it my intention to build a system to recognize them.

An observation platform was devised to show moving sequences of a person gesturing with her head at different rates. Observers were asked to determine, for each sequence, if the gesture looked natural, too fast or too slow. Each observer saw three repetitions of each gesture, played at eight different speeds. The speeds ranged from approximately 1 gesture every 3 seconds, to approximately 3 gestures per second. The sequences were shown in random order.

Table 1 and the accompanying graphs show the results of the observations. The table shows that most people accept gestures at a rate of 1 gesture per second to be the most natural. This is the desired rate for the head locator to operate. Detecting gestures is a natural extension to the head locator. If this is to be accomplished, it will have to operate at 2 frames per second, which is the Nyquist frequency for head gestures.

Gesture 1. Shaking the head "No"				
Gestures/Second	Yes %	Unsure %	No %	Mean Response
3.73	8.77	17.54	73.68	-0.49
2.82	17.24	22.41	58.62	-0.28
1.89	51.72	25.86	20.69	0.33
0.94	87.72	10.53	1.75	0.75
0.94	85.96	12.28	1.75	0.74
0.46	57.89	22.81	19.30	0.26
0.31	14.04	28.07	57.89	-0.46
0.20	7.02	19.30	73.68	-0.63

Gesture 2. Nodding the head "Yes"				
Gestures /Second	Yes %	Unsure %	No %	Mean Response
2.80	19.30	24.56	56.14	-0.25
1.86	61.40	19.30	19.30	0.37
0.94	82.46	12.28	5.26	0.61
0.94	80.70	15.79	3.51	0.63
0.47	36.84	29.82	33.33	-0.02
0.32	15.79	24.56	59.65	-0.42
0.25	5.26	8.77	85.96	-0.72
0.20	3.51	7.02	89.47	-0.72

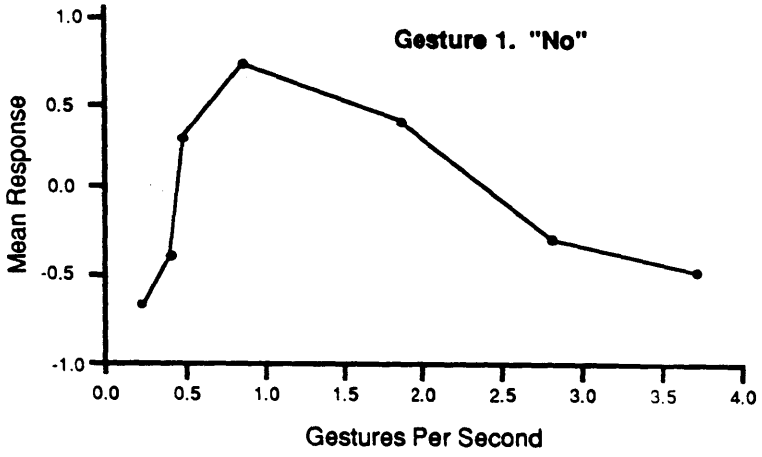
Experiment summary:

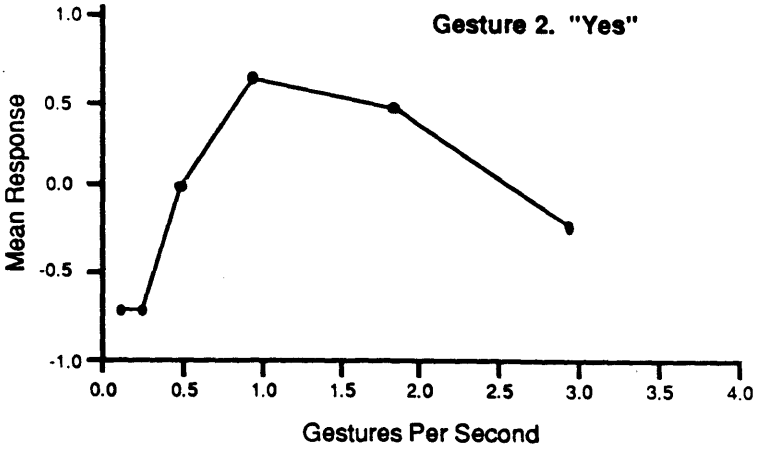
18 Subjects were shown 2 gestures at 8 frequencies. Subjects were shown gestures 3 times at each frequency. The order in which the gestures were shown was randomized for each subject. The "Mean Response" column the mean of the yes/unsure/no responses for each speed, where "yes" is counted as 1.0, "unsure" is counted as 0.0 and "no" is counted as -1.0. The frequency recorded in the first column is the mean frequency over all trials.

Table 1. Head Gesture Timing Experiment

5.3. Tools

The head server is implemented on a Sun workstation equipped with a Data Cube frame buffer which can digitize frames from live video input. Live video is fed to the Data Cube from a Panasonic video camera with a wide-angle lens. Figure 5.1 shows a block diagram of the system setup. For a complete description of the hardware configuration, see Appendix B.





Graphs 1 and 2 correlate gesture speed with a subjective measure of "naturalness"

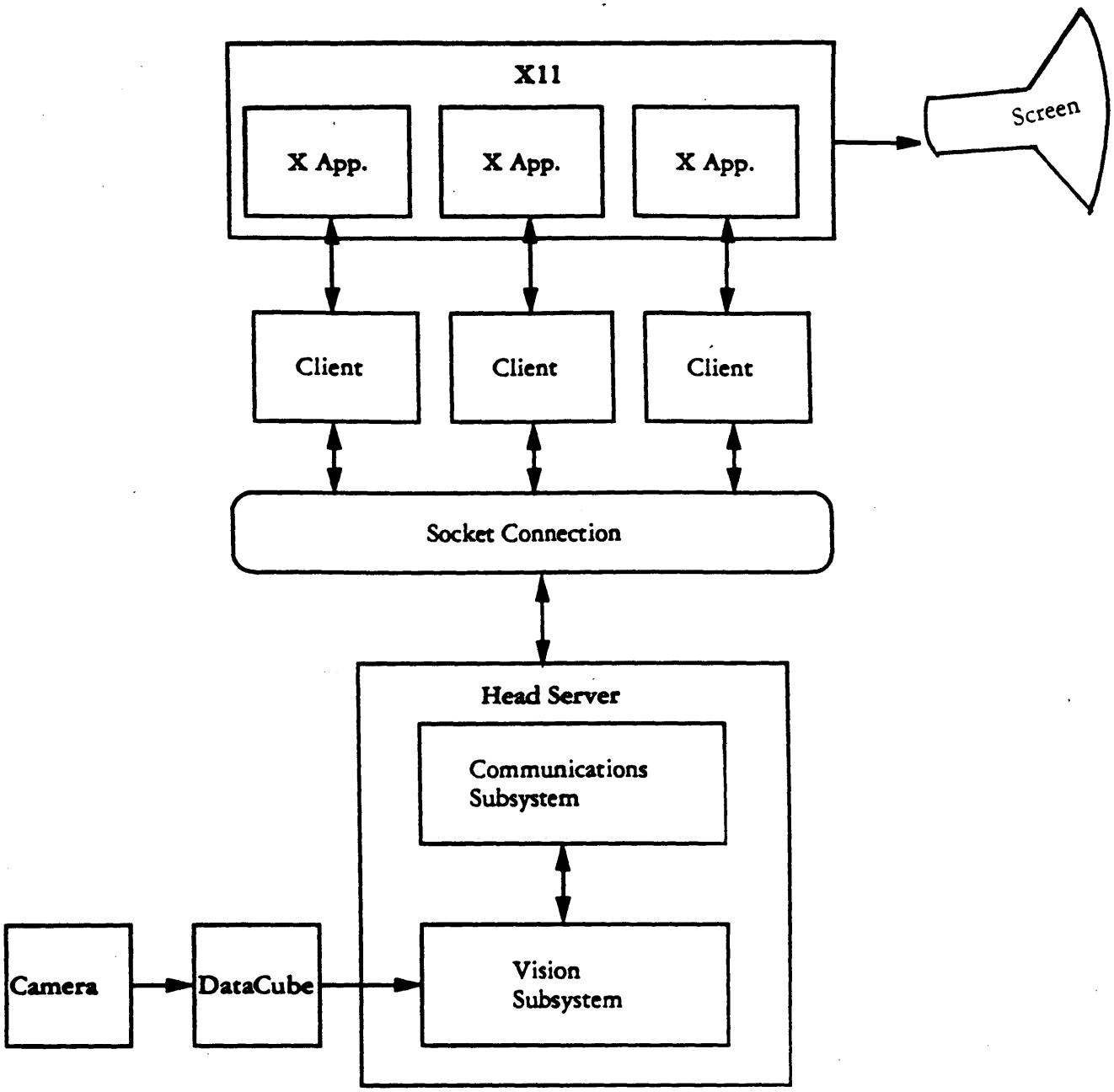


Figure 5.1. Head Server Block Diagram

The clients and server communicate through a *socket*, which is a Unix construct that can be thought of as a shared file. The name of the socket is known to the server as well as all of the clients. Data that is written to the socket is available for reading by all of the processes that have the socket open. Sockets can also be accessed across file systems via a local area network so that the head server may reside on a separate computer from its clients[7].

X Windows is a device independent window oriented interface that consists of a server, a protocol, and some number of clients. X Windows makes a distinction between a *Screen* and a *Display*. A Screen is a physical entity, a CRT. A Display is a single computer which may have more than one Screen attached to it. The X server is located on the machine that supports the display, but not necessarily the same machine as the client. The X server is the only part of the system that draws directly to the display. X Clients communicate with the server using a fixed protocol[8], through a socket.

A typical X Windows screen looks like the one shown in Figure 5.2. There are typically several kinds of clients running simultaneously. Some, but not necessarily all of the clients accept keyboard input. Input is directed to a window by positioning the cursor inside its boundaries, or by positioning the cursor and then clicking the mouse button. All of the items displayed on the screen can be moved, resized or deleted by manipulating the mouse and keyboard.

---

[7]As of the current implementation, the system uses a Unix domain socket, which only accepts connections on the local host. Subroutines are available to allow communication through a TCP socket, which would allow the server to communicate with clients on distant machines.

[8]X11, Revision 4.

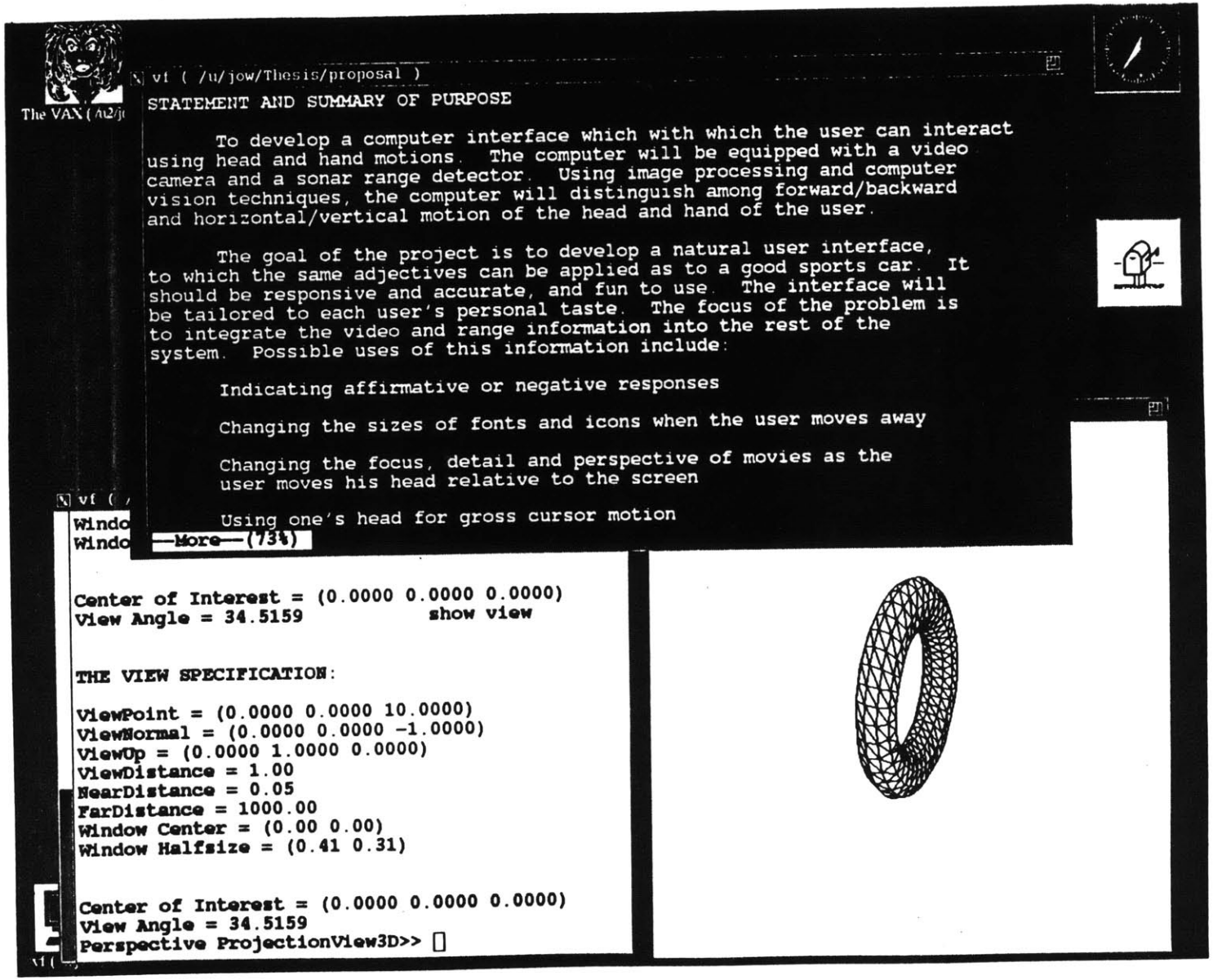


Figure 5.2. An example of an X Windows display

The exact behavior of an X Windows session is controlled by a Window Manager, which determines the appearance of the windows, their reactions to cursor movements and buttons (for example, whether or not it is necessary to click the mouse button in order to redirect keyboard input), their initial placement on the screen, and their interaction with other windows. In addition, most Window Managers provide a set of menus with which the user can change the the way the screen looks, issue commands to initialize or kill clients, or reconfigure the Window Manager itself. The Window Manager is largely responsible for creating the "personality" of the interface.

5.4. Client/Server Communication

A very simple protocol is used between the head server and client processes. The conversation between the two follows a simple pattern in which the client asks the server for information about the location of the user, and the server responds a short time later.

5.4.1. Opening a Client-Server Socket Connection

The client establishes a connection to the head server socket by issuing a library call. This call will fail if the head server is not available, or if a connection to it could not be established. If the call is successful, a unique connection id number is returned. This id number is used in all further communication between the client and the server. The connection id specifies a connection between the client and the server in the same way as a file descriptor specifies a channel through which to communicate with the file system.

5.4.2. Request Packets

Information is exchanged between the client and the server in packets of information called *Requests*. A Request is a data structure which contains a *request type* and a *data packet*. The request type specifies what kind of information is contained in the data packet (if any). The data packet can contain up to 110 bytes of information.

When the client is ready to request information from the server, it issues a library call that blocks (waits) until the server has processed the request. A pointer to a request packet is returned by the system. The request type of the returned packet contains confirmation that the request was properly processed, or an indication of the type of error that occurred. The data portion of the returned packet contains the requested information.

There are currently five client requests that can be sent to the server. The meanings of the requests are:

- send all information about the head
- send only the distance from the screen to the head
- send the name of the active window
- change the name of the active window
- disconnect this client from the server

The Client-Server conversation continues indefinitely until the client issues a disconnect request, or until the server is shut down. For detailed information about the Client-Server protocol, see Appendix C.

5.5. Head-Server Vision Subsystem

When a client request is received by the server, the request type is decoded and dispatched to the proper processing routines. Client requests about head information are

passed directly to the vision subsystem. Other requests are handled by internal service routines. When the request has been processed, the server assembles a new request packet containing the requested data (if any) and a return code indicating success or failure. The new packet is sent back through the communications channel to the requesting client.

5.5.1. Design considerations

The head server is not concerned with the color, size or hairiness of the person using it. Features such as these could be useful in helping to identify the user. However, it was beyond the scope of the project to build a vision system that was smart enough to tell the difference between unusual features and noise. Therefore, the system looks for properties of the human face that are independent of skin and hair color, relative size of features, and whether or not the user has facial hair. To this end, the system seeks out the contour of the silhouette of the user's face, because the shape of a human face is similar enough from sample to sample to make reasonable generalizations about the identity of features.

Leonardo Da Vinci shows the proportions of the human face in Figure 5.3 [Pedretti 1977]. The fact that these proportions hold true among a variety of individuals is useful, because the vision system needed to make some assumptions about the placement of features. For example, the distances from *b* to *c*, from *c* to *f* and from *f* to *i* are all $\frac{1}{3}$ of the face. The eye must fall somewhere between *c* and *f*, regardless of the angle of the face. Furthermore, the nose (at *f*) is a "large bump" somewhere in the second third of the face. The only proportion the system needed that Leonardo did not provide was the depth from the bridge of the nose to the bottom of the eye socket. The program assumes this to be the same as the amount that the nose extends from the face, but that number can be adjusted on an individual basis.

5.5.2. Finding the Position of the Head

Viewing the head in silhouette exploits the fact that it is almost always possible to find the three most prominent features of the face. The nose is the largest bump on the face. Even in a silhouette of a delicate face, the nose is an unmistakable protuberance. The chin easily identified, because the space under the chin retreats deeply towards the neck. Under average room lighting conditions, the eye sockets are in shadow, and thus constitute a third recognizable feature.

The head is silhouetted against a solid background. The camera is aimed so its line of sight is perpendicular to the computer screen, so that a person staring straight at the screen will appear in profile. A black sheet is draped behind people who have a light complexion to create a dark background. For people who have a dark complexion, a white sheet is used. The point at which it becomes necessary to change from a black to a white background depends on a programmable threshold value.

The photosensitivity rating on the camera is 0.3 lux, which is very sensitive, so no special lighting is required. The head is illuminated by a common Luxo™ lamp fitted with a standard 75 watt light bulb and covered with a piece of paper to enhance diffusion. Ambient lighting comes from several fluorescent ceiling lamps.

The profile of a face actually contains several small features, including the bridge of the nose, the lips, and the indentation above the chin. In addition, there can be many more small discontinuities in the slope of the profile, owing to facial structure or disturbances on the surface of the skin. In order to ensure that the head locator is sensitive only to the

large features, the profile is low-pass filtered before it is processed. This is done by unfocussing the lens.

The captured video is thresholded, and from then on treated as a binary image. The program scans from the top right to the bottom left, searching for the edge of the face. When it is found, the contour of the face is extracted by scanning downward until the tip of the nose is located (Figure 5.4). Using this method, the program is not disturbed by moustaches or beards, which, on light skinned people, will appear the same color as the background.

Once the nose location is known, the program scans leftward and upward, using the face contour as a boundary, to search for the eye. The eye socket appears as a large patch of black somewhere between the nose and the forehead. The distance from the eye socket to the front of the face is used to calculate the amount of Pan. The eye socket shows up as a dark spot in the middle of the profile because it is a depression that causes a shadow. The distance between this spot and the edge of the profile changes as the head moves to the left or right.

The vision subsystem is able to locate the nose and eye in approximately 1 second. If no dark patch can be found, the system reports that the eye is not visible. If the face contour does not reveal a bump that could possibly be a nose, the system returns a failure code. Further discussion of modes of failure and their meaning can be found in Chapter 7.

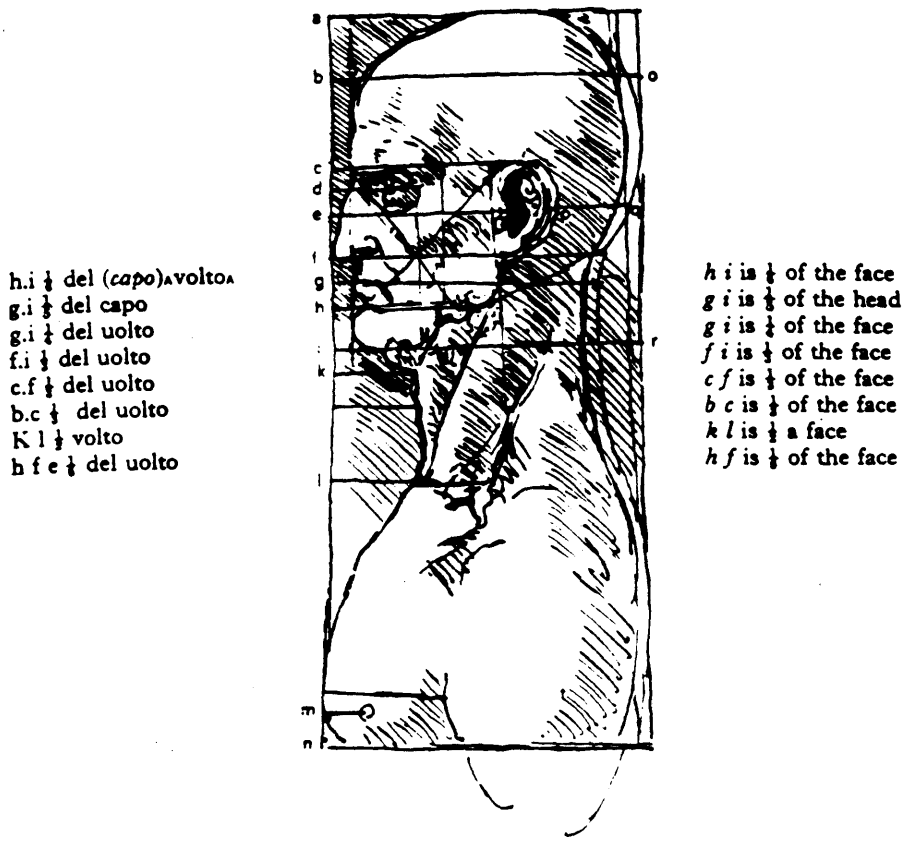


Figure 5.3. Proportions of a human head drawn by Leonardo [Pedretti 1977].

5.5.3. The Head Measurements

The purpose of the vision subsystem is to find four parameters: *XDist, YDist, Pan* and *Tilt.* XDist is the distance from the tip of the user's nose to the front of the workstation screen. YDist is the distance from the top of the user's head to the top of the field of view of the camera. Pan is the angle formed between the user's line of sight and the horizontal center of the screen. Tilt is the angle between the user's line of sight and the vertical center of the screen.

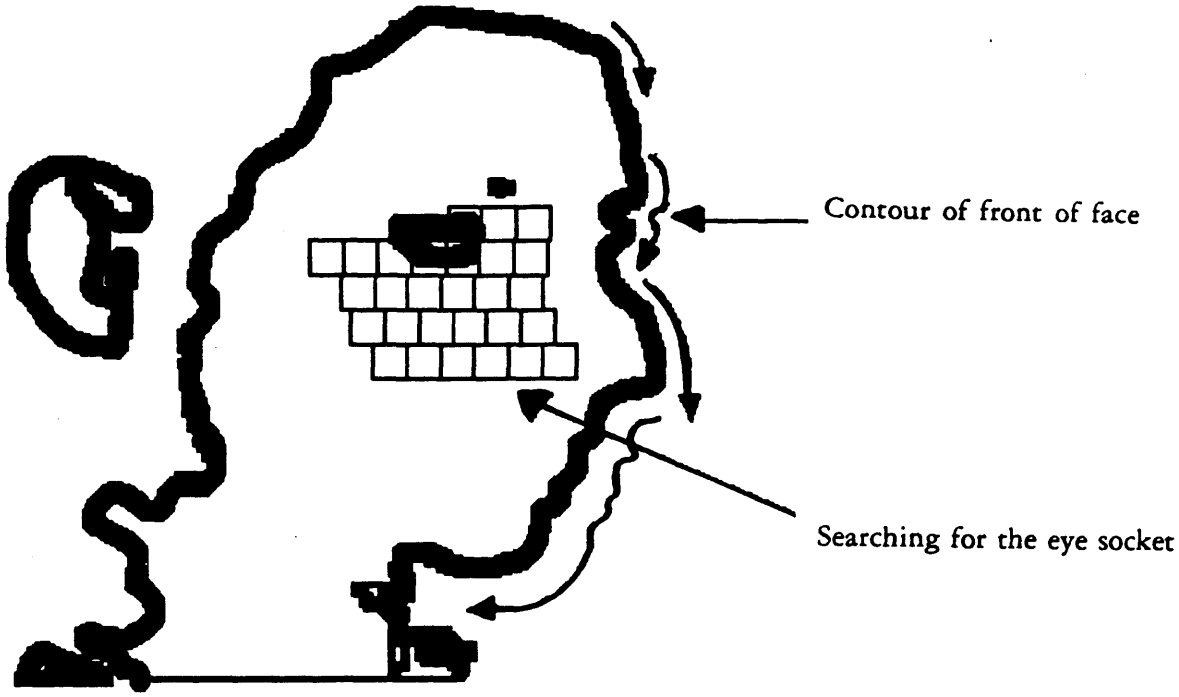


Figure 5.4. Finding the face contour and the eye socket.

Finding XDist and YDist

XDist and YDist are taken directly from the digitized image, measured in pixels from the face to the edge of the image. A conversion from pixels to centimeters can be calculated, if the distance from the camera to the user is known. XDist and YDist in centimeters, are given by:

$$XDist = \frac{X\partial\cos\alpha}{u} \qquad (F5.1)$$

$$YDist = \frac{Y\partial\cos\alpha}{v} \qquad (F5.2)$$

where:

∂ = distance from camera to user, in centimeters,

α = half field of view of the lens, in degrees,

X = pixels from user to right edge of image,

Y = pixels from user to top of image,

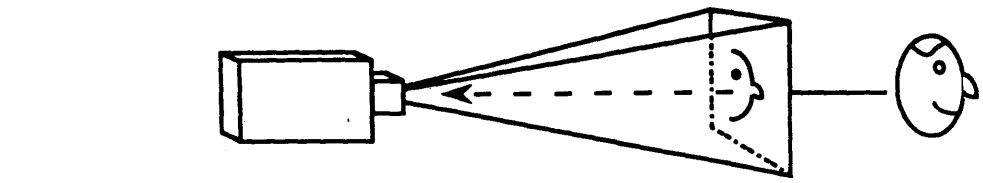
u = $\frac{1}{2}$ width of digitized image, measured in pixels,

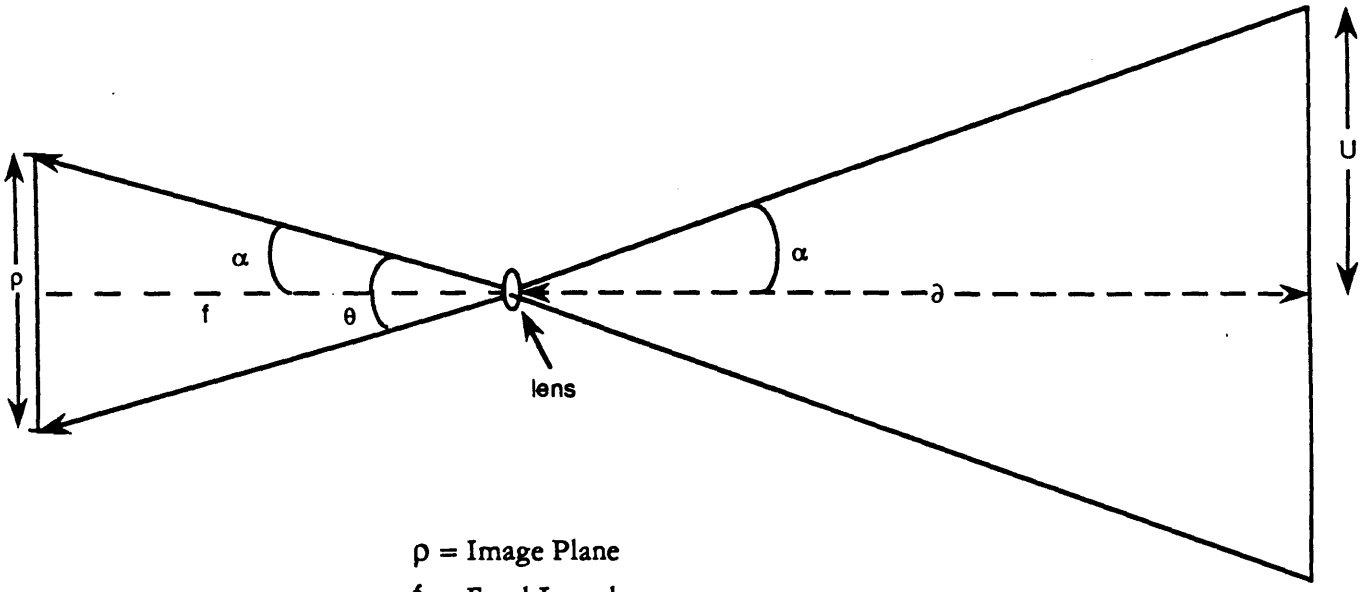
v = $\frac{1}{2}$ height of digitized image, measured in pixels.

The half field of view α of the lens is calculated from the focal length, f and the image plane size, ρ:

$$\alpha = \frac{1}{2}\theta = \arctan(\frac{\rho}{f}) \qquad (F5.3)$$

Figure 5.5 illustrates the geometry of the camera's field of view.





ρ = Image Plane

f = Focal Length

∂ = Object Distance

$$\frac{\text{Field of View}}{2} = \alpha = \tan^{-1}\left(\frac{\rho}{f}\right) \quad \text{[degrees]}$$

$$\frac{\text{Field of View}}{2} = U = \partial\cos(\alpha) \quad \text{[meters]}$$

Figure 5.5. Camera field of view

Finding Pan

Pan is a function of the distance between the center of the visible eye socket and the bridge of the nose (Figure 5.6). The distance that represents 0° of pan is set on user-by-user basis. The angle of pan is limited to 30° in either direction. If the user turns away from the camera more than that amount, her eye socket becomes invisible. If she turns towards the camera more than that amount, the face no longer appears as a profile, and the vision algorithms degrade. The formula for calculating the angle of Pan, therefore, is:

$$P = 30 + \frac{E_b - E_c}{E_c} \qquad \text{(F5.4)}$$

where:

P = angle of pan,

E_b = distance from eye to bridge of nose, and,

E_c = preset eye/bridge distance for centered gaze.

Finding Tilt

Tilt is the angle of the front of the face, with respect to horizontal. It is measured using the line segment from the tip of the nose to the forehead (see Figure 5.7). Tilt is calculated simply by converting the slope of that line to an angle, by:

$$T = \arctan\left(\frac{F_y - N_y}{F_x - N_x}\right) - C_t \qquad \text{(F5.5)}$$

where:

F_x, F_y = coordinates at forehead,

N_x, N_y = coordinates at tip of nose, and

C_t = calibrated constant for centered gaze.

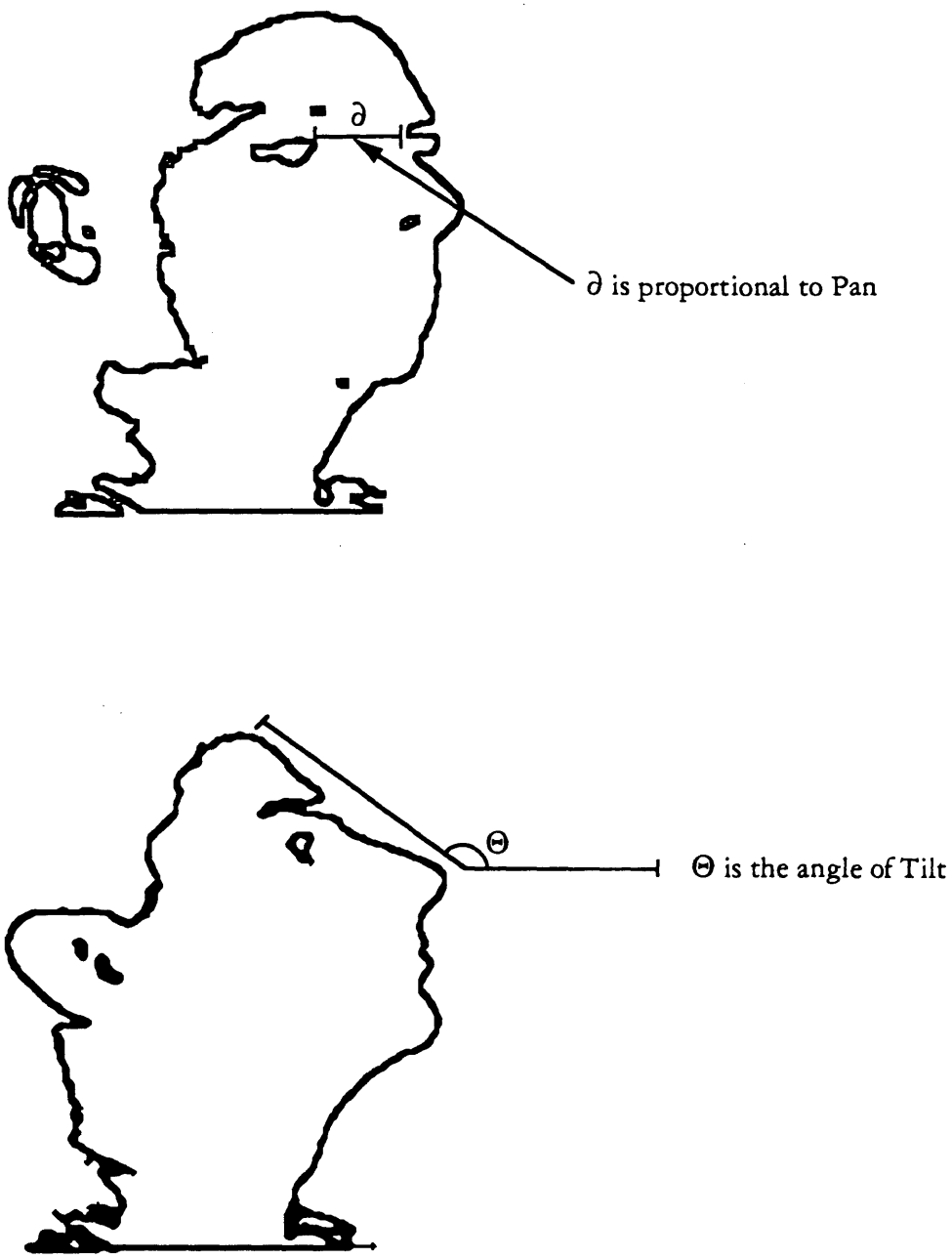


Figure 5.6. Measuring Pan and Tilt.

Chapter 6. The Client Applications

Several applications were developed to demonstrate the operation of the head server. Individually, the applications are interesting novelties. When they are all run simultaneously, however, the user begins to feel that she is using something approaching an autonomous interface. The applications were chosen to encourage that impression when they act in conjunction.

6.1. Presence Detection

The Presence Detector is the simplest of all of the applications. The idea is based on a project by [Gulsen 1988], which used a video camera equipped with a connection between the autofocus lens and the serial input port of the computer. In the original project, the appearance of any object in the scene caused the autofocus lens to rotate until the object was in focus. By measuring the amount of rotation, the computer was able to calculate the distance between the object and the camera. The fact that the lens rotated was an indication that something had moved along a Z axis, relative to the camera. An assumption could be made that, if nothing moved for a certain period, the user was not present. This was only an assumption, though, and was not, as such, reliable.

The presence detector developed in conjunction with the Head Server knows immediately if anything is in the scene, because the scene always consists of a solid background plus the user. If the image from the video camera is empty, nothing is present in the scene.

At the moment, the Presence Detector is only used to toggle the screen saver mode in X11. If the user leaves the work area, the screen saver is turned on. When she reappears, the screen saver is turned off. There are other things this application could do. For example, when the user goes away the computer could devote more time to background tasks than to the user interface. When she returns, any new mail or completed background processes can be reported, and the priority of the interface software can be raised again.

6.2. Resizing the active window

Another feature of Gulsen's Presence Detector was that it would automatically enlarge or reduce the size of the window in which the program was running, as the user moved further from or closer to the screen. The intention was to give the impression of a window that maintains a constant size, relative to the user's field of view. A modification to the original program, called OTerm, also changes the size of the font that appears in the window.

A similar client was developed for the head server. The user chooses which window is to be affected. When the user moves closer to the screen, the window is shrunk and the type gets smaller. When the user moves further away from the screen the window and type are enlarged. Owing to the limited availability of font sizes, it was not possible to build the program so that the window subtends a constant field of view. Font sizes were chosen on the basis of their legibility at various viewing distances.

It is not absolutely clear whether resizing the active window is a desirable thing to do. On one hand, programmers tend to lean forward and backward in their chairs over time while working with the same window, so it is a useful feature. On the other hand, when she leans away from the screen, the user may wish to see more of the action. It might make

more sense for all of the windows to get smaller in that case, so that they no longer overlap and the user can see everything at once.

6.3. Selecting the active window

One of the more frustrating properties of a typical window system is that it is necessary to manipulate the mouse in order to indicate which window is to receive keyboard input. When using a multi-window environment, the user's attention shifts among windows quite frequently, making it necessary to handle the mouse approximately 20% of the time. In environments where there are several screens, the difficulty of tracking the cursor is increased, because the window space is not physically contiguous.

In order to alleviate this problem, a client was developed that activates the window closest to the user's line of sight. (The head server provides a simple function call that provides the coordinates of the intersection of the user's line of sight with the screen.) In the event that that point is enclosed by a window that accepts keyboard input, the window becomes the active window. This means that, by moving her head so that her line of sight appears to intersect the desired window, the user can control which window is selected for keyboard input. An easy way to think about this is for the user to aim her nose at the desired window. For systems employing more than one screen, this is a very natural thing for the user to do. When she turns her head to look at a particular window, the destination of keyboard input follows her gaze. On systems with small screens, the head motion needs to be a bit contrived. This application would be ideal, however, in systems that use a very large screen (such as air traffic controllers), or that use multiple screens (the stock exchange, nuclear power plants, television studios, etc.)

6.4. Warping the Cursor

As an experiment, a client was built that moved the cursor so that it always appeared where the user was looking. This turned out to be a bad idea because of the Midas Touch problem, and because the user expected to find the cursor where she left it. The client moved the cursor wildly about the screen causing a great deal of confusion. In order to alleviate this problem it should be possible for the user to indicate that the cursor is to be moved to the place where she is looking by clicking on the mouse or typing a key.

This entire problem goes away, however, if the interface is built without cursors and without points of control. Once the computer can identify the active window, the user will need to manipulate the mouse approximately 20% less, anyway, so perhaps the mouse becomes less of a necessity.

6.5. Controlling Digital Movies

Because the head server reports the location of the user's head in three dimensions, it encourages us to think about an interface *space*, rather than just the surface of the screen. For example, the user may explore the $2\frac{1}{2}$D space occupied by layered windows in a window system by tilting her head or turning in such ways to indicate looking "behind" windows, or exposing hidden ones. At the moment, the head-server is not a substitute for a Head Mounted Display or the kind of setup used in Artificial Reality systems such as those designed by [Zimmerman 1987]. It does, however, change the way we can think about interface space.

Many computer systems display digitized movies on the screen in addition to text and graphics. The movies are usually presented as a fixed sequence that was recorded from a single camera angle. With proper image coding techniques, however, it is possible to allow the user to gaze *into* the movie, in effect changing the apparent camera angle.

The client that demonstrates this presents a moving sequence on the Datacube display. The user can control the point of view of the "camera" by changing her position relative to the display. In order to see parts of the scene that are off the screen, the user may turn her head left or right, or tilt it up or down. This causes the scene to scroll in a manner similar to the scrolling of a text window. The visual effect is similar to the cinematic effects called *dollying* and *craning.* A *dolly* lateral motion of the camera, caused by moving the camera assembly along a track. The camera itself remains stationary, but its point of view changes horizontally. In a *crane* shot, the camera is mounted on a crane and can be moved in any direction. In this case, the user can simulate a crane shot in which the camera is moved vertically only.

When the user moves closer to the screen, the movie is enlarged, showing a smaller area in greater detail. This is similar to the cinematic *zoom* effect.

These effects are accomplished by working with an original that is twice the area of the frames that are displayed. A subsection of the decoded movie is extracted and displayed on the output device. The dolly and crane effects are accomplished by displaying different subsections of the sequence (Figure 6.1.)

The zoom effect is made possible by an image coding technique (subband coding) which stores the sequence as a collection of separate images, each containing a different amount of visual detail. The levels of detail are called "subbands". The decoder can be instructed

to construct each frame by extracting and summing any number of subbands. The number of subbands that are summed corresponds to the total amount of detail seen the completed frame (see Figure 6.2).

Although the client applications demonstrate some of the capabilities of the head server, they by no means constitute an exhaustive demonstration of its possible contribution to the development of an autonomous, intelligent interface. The current implementation has many drawbacks, as will be explained in the next chapter. However, the clients that have been built do significantly enhance the experience of using a window system by providing a quality of system response that had not previously been available.

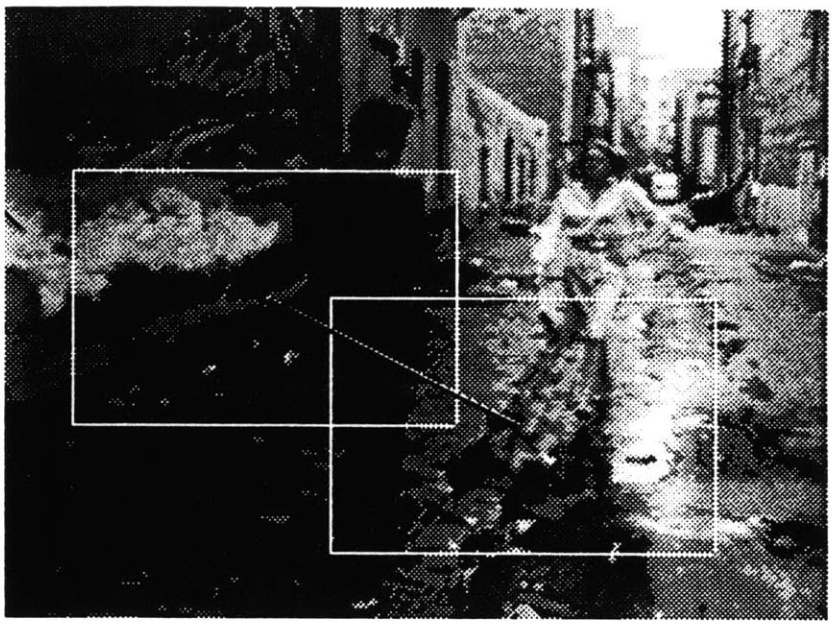


Figure 6.1. The cinematic effects "dolly" and "crane" are achieved by changing the origin of the frame that is extracted from the original sequence.

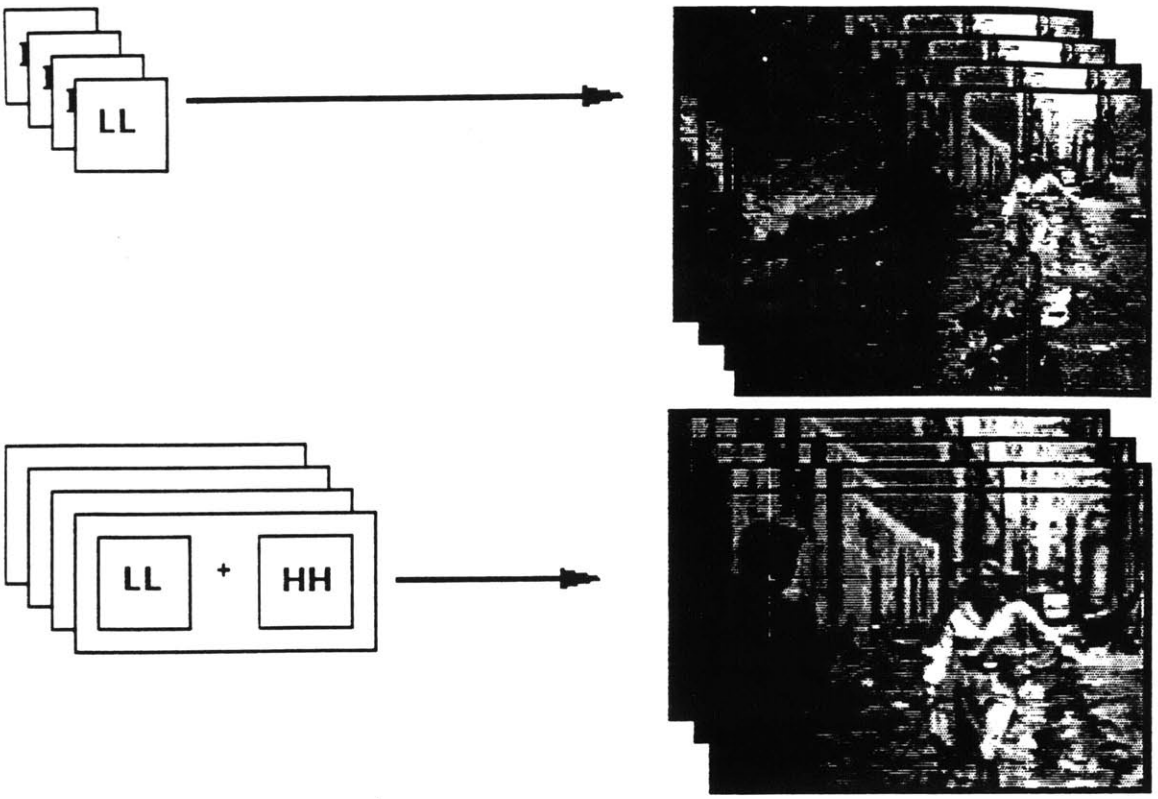


Figure 6.2. An illustration of subband coding. The sequence containing low frequencies only (LL) has less detail than the sequence containing high and low frequencies (LL + HH).

Chapter 7. Evaluations

The addition of a head locator to a system is not sufficient to create an intelligent interface. As is the case with any input device, the head locator makes certain information available, but does not give hints about how that information should be used. An intelligent interface is the product of a well considered design and the available technology. What can be said, however, is that the development of an intelligent interface becomes a greater possibility with the invention of devices such as the head locator, that provide information about the posture of the user.

7.1. Evaluation of the Vision Subsystem

It is important to note that this system is not intended to be a landmark in research in computer vision. In relation to the size of the entire project, little work has been done to make the vision subsystem extract more than minimal information about the position of the user's head. Certainly, a more spectacular demonstration would arise if the system were able to recognize individual users, or to pinpoint exactly where the user were looking. Such a system is beyond the scope of this thesis and, to a certain extent, is more appropriately addressed by those engaged in computer vision research.

The current implementation is by no means the only possible one. The Head Locator bears close resemblance to the Realtime Headreader system developed by Mase, Watanabe and Suenaga at NTT [Mase 1990]. It should be recognized that the Headreader has been in development at least since 1988, and has only recently reached its current level of accuracy and reliability. For background discussion of the Headreader system, see [Mase 1987].

The operation of Headreader and Head Locator are different in several respects. The Headreader attempts to locate the centroid of the head and face, and a rotation point on the neck. The user is oriented face-on, with respect to the camera, so that her nose is pointing towards the lens. The system attempts to recognize what I have called "tilt" by measuring the gradient of the light reflected off of the head as it is tilted forward.

Both the Headreader and the Head Locator exploit the contrast between face and hair, against a known (or blank) background. It is interesting to note that Mase writes, "the face contour search proceeds from bottom to top of the entire head region, because the lower face area is less affected by hair or eyebrows". In contrast, I designed the Head Locator to scan from the top of the head towards the bottom, in order to allow for bearded or mustachioed users. The presence of a beard or moustache will cause the Head Locator to believe it has reached the chin, or the bottom of an exceedingly large nose. The system does not fail in either of these cases.

The Headreader is able to detect orientation and motion in approximately real time. Mase reports frame rates of 5-10 fps, while the Head Locator is able to detect head orientation at about 1-2 fps. The difference in speeds is probably explained by the fact that the Headreader runs on a Sun4/260C, and processes a 128x120 pixel image. The Head Locator runs on a Sun3/60, and, at the default subampling setting, processes a 320x240 pixel image.

7.1.1. Advantages

Certain features of the head locator have worked out particularly well. Describing these features might prove useful to anyone who wishes to engage in similar work in the future.

The vision subsystem does not use any sophisticated image processing algorithms. Finding a contour against a blank background is a relatively simple task. Despite the fact that Fourier analysis or depth-from-stereo/depth-from-focus programs might provide more accurate information about the shape of the head, these methods are much more complicated than is really necessary.

The system works with a wide variety of faces. Because of the various adjustable thresholds, the system works equally well with dark skinned people as with light skinned ones, with bearded users or clean cut ones, and with people who have delicate faces as well as people who have dramatic features. Once thresholds are set for a particular face, they can be stored and used again. This means that an interface can be programmed with individual preferences that are activated when the thresholds are recalled. When computer vision systems advance to the stage where they can recognize individuals, the preferences can be set automatically. This combination of factors will give the user the impression that the system knows her individually, which is one of the requirements of an intelligent interface.

7.1.2. Accuracy

Each value returned by the head locator is accurate to within a different tolerance. Tolerances were measured by observation. The most accurate values are XDist and YDist, because they correspond to absolute measurements of pixels.

The Tilt value has an error tolerance of approximately ±10%, because the point chosen on the forehead from which to measure the nose-forehead angle is defined to be the first pixel found in the scanning process. This point changes slightly as the user turns her head.

The Pan value has an error tolerance of approximately ±20%, because of the inaccuracy of the relationship between the eye-nose distance and the actual turning angle.

The result is that the user can activate any point on the screen, within an approximate ±30% margin of error, by pointing with her nose. This is adequate for selecting from up to three large windows. Unfortunately, most users who are programmers have five or more windows open at a time (as observed in Chapter 1), which renders the system useless for that kind of work.

7.1.3. Disadvantages

There are two fundamental disadvantages of the current implementation of the head locator. It cannot identify the actual line of sight, and it is not fast enough to allow gestural input.

The eyes and the head move somewhat independently. Therefore, it is impossible to identify the actual line of sight from the position of the head. In order to do this accurately, an eye tracker must be used. This deficit means that the head server cannot be used as a replacement for a mouse.

7.1.4. Posture Restrictions

Owing to the nature of the vision subsystem, the user's eye socket and nose must be visible to the camera at all times in order for the head locator to work. This places restrictions on the user's movement. The user's head must remain within a small volume. She must not turn her back to the camera, nor may she face the camera directly. In both of these cases the profile is lost and the vision subsystem fails. In addition, the system is confused if anything except a facial profile is visible in the scene, which means that the user must keep her hands and other objects out of the field of view of the camera.

Full Motion Volume

The volume to which the user is confined, if she wishes the head locator to work, is determined by the distance between the camera and the user, the camera itself and the kind of lens being used. The current configuration uses a 2/3" Vidicon tube, and a 25mm lens. The camera is placed approximately 1 meter from the user. This distance is only practically limited, insofar as it is good to have the camera close to the user so that it may be properly adjusted. Using the formulas in chapter 5 we can find the plane subtended by the camera's field of view.

The field of view θ lens is calculated from the focal length and the image plane size:

$$\theta = \arctan(\frac{25.0}{16.9}) = 55.9^\circ \qquad \text{(F7.1)}$$

The object plane width is:

$$O_W = \partial\cos(\theta) = 0.56 \text{ meters} \qquad \text{(F7.2)}$$

The object plane height should be the same, because the lens is circular. However, the aspect ratio of the Datacube display is $\frac{768}{512}$, or $\frac{1}{0.66}$, so the object plane height O_H is 0.36 meters, at a camera distance of $\partial = 1$ meter.

Head Motion Volume

The motion of the user's head is restricted to a narrow range. The total pan motion allowed is 60°. If the user turns towards the camera more than 30°, the profile is lost and the vision subsystem fails. If the user turns away from the camera more 30°, the eye socket can no longer be found. Although people can turn their heads almost 180°, in a typical computing environment it is rarely necessary to do so. The restriction to 60° is not prohibitive.

The tilt motion of the user's head is not restricted, assuming the user does not bend forward or backwards. The human head can normally be tilted about 45° by the neck alone. The vision subsystem can extract the contour of a silhouette within a 180° tolerance. Figure 7.1 illustrates the practical operating volume of the head server.

7.2. Evaluation of Applications

Because an engineering model of user performance with or without the head locator (á la Card, Moran and Newell) is beyond the scope of this thesis, and because the head locator still only a research prototype, analyses of the applications on an individual basis will suffice. I will only consider the three applications that directly affect the appearance and control of the interface.

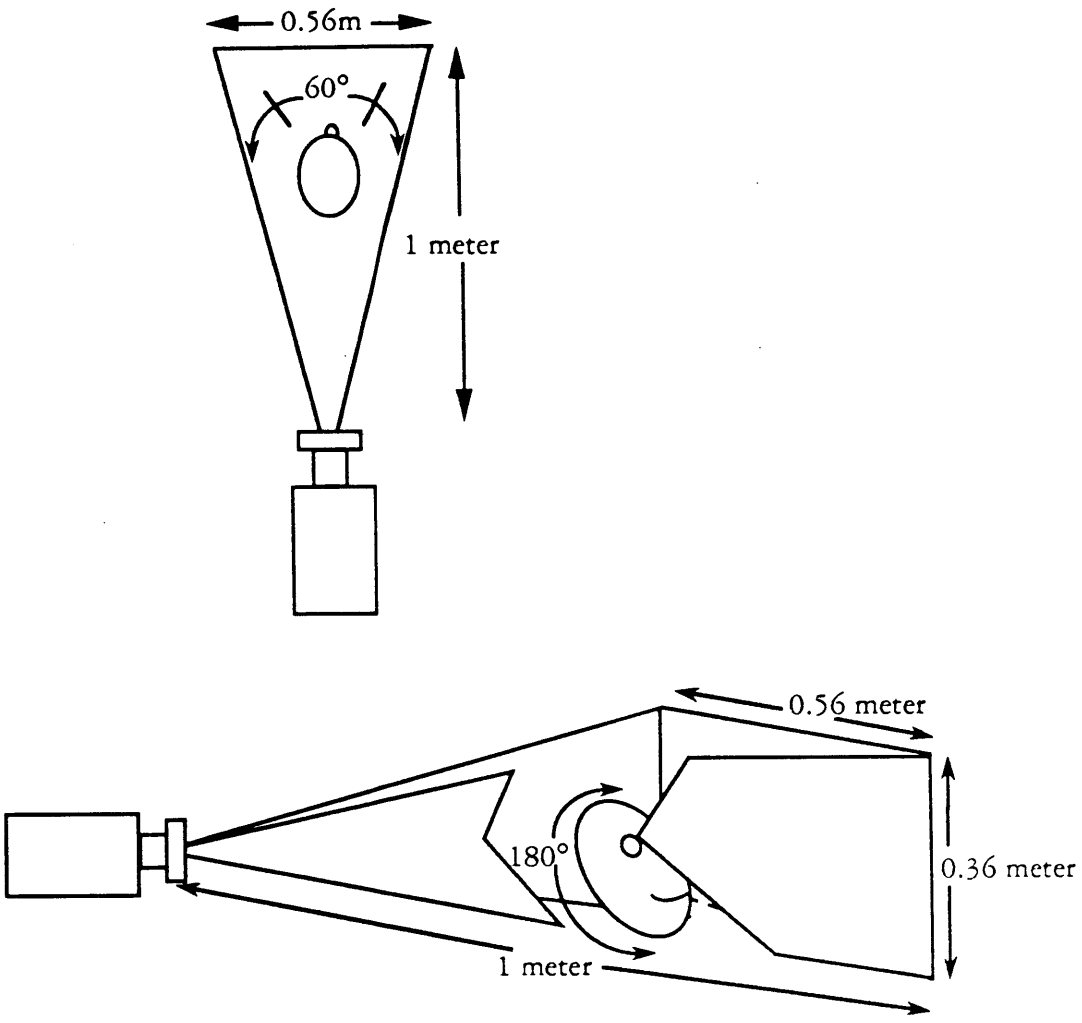


Figure 7.1. Viewing Volume

7.2.1. Presence Detector

The presence detector is the most accurate of the applications that have been developed so far. The presence or absence of the user reliably causes the screen saver to toggle within 1 second of her arrival or departure.

7.2.2. Resizing the Active Window

Users report that this is an intriguing, but slightly annoying feature. It is annoying partly because the screens change their size in large quanta, rather than continuously. In addition, people are not accustomed to interfaces that act in this manner, and so the novelty is a distraction. As mentioned above, it is not clear whether it is always desirable to resize windows, especially if the text is not sized proportionally to the user's distance. However, as a tool for a programming environment this is sometimes a nice feature, especially if the user wishes to lean away from the screen while she edits her text without straining her eyes. In any case, this application gives the user the sense that she is being watched, that the computer has some idea about what she is doing.

7.2.2. Identifying the Active Window

The only drawback to this client is that it is not always accurate. The inaccuracies inherent in the vision subsystem are compounded when the available windows are smaller than the system's tolerance. The resulting symptom is that the wrong window is activated, or no window is activated. However, the application works quite well when there are two or three large windows on the screen. When this configuration is obtained, the user can switch among windows by aiming her nose at the one that is desired.

It is not possible, as of this implementation, to automatically raise windows that are completely covered by other objects on the screen. The application only works on windows that are partially visible when they are not active.

7.3. Improvements

The role of the head locator as a component of an intelligent interface can only be fairly evaluated if the error tolerances are narrower, and if it allows more freedom of movement on the part of the user, especially if it does not prohibit her from placing her hands near her head. I have contended in this thesis that a perfect incarnation of the head locator would be an invaluable component of any interface that purports to know its user. The present implementation of the head locator could be greatly improved by applying more sophisticated computer vision techniques, and by running it on a more powerful platform. More powerful platforms are already available. As a discipline, computer vision is in its infancy. It is assumed that the requisite advances will be made within a few years.

One more remark is necessary regarding the usefulness of the head server in an X Windows environment. X Windows does not in any way resemble an intelligent or autonomous interface. It is not likely that it can be modified to do so. Providing X Windows with head server information might improve its "look and feel", but it retains all the drawbacks and inconveniences of a window system. The ideal environment for the head server would be an autonomous interface, which I hope will be developed in the future. As far as the current implementation is concerned, it should be seen as a cairn, not a milestone.

Chapter 8. Conclusion

In this thesis I have presented a novel input device, the head locator, that uses simple hardware and software to provide the user interface with information about the orientation of the user's head. I have discussed the way that this device fits into the general problem of user interface design. Some demonstration programs were developed, and their operation was evaluated in terms of an ideal called the "intelligent interface".

This thesis has been consistent in its philosophy that, as a rule, computer interfaces should require very little specific instruction from the user. The head locator was developed in order to show that, in addition to mouse motion and keyboard input, there are additional relevant cues that the user can provide. By building a device that recognizes some of those cues, the human interface is enhanced.

8.1. Future Work

I recognize that the head locator is far from complete, and I hope that the present work inspires further research. Avenues for future work require the development of systems that can take advantage of better solutions to the problem of computer vision, and that require less time per frame to run. It seems proper that the goal is to build systems that can communicate with people in several modes--verbal, visual, gestural, and tactile--since these are the modes in which people communicate with each other.

8.1.1. Gestural Input

Head gestures (nodding, shaking, shrugging) are useful but limiting insofar as they can only answer binary questions. The Headreader system can recognize simple head gestures. With very little improvement, the it can probably recognize more complex ones. Some examples are: nods and shakes of various amplitudes, because amplitude corresponds to the reliability of the answer; shrugging the shoulders, which indicates indecisiveness; swinging the head quickly towards a target, to indicate the intended motion of one object towards another.

Research is currently being carried out on the use of hand gestures using a Data Glove [Sturman 1989]. It would be exciting to see a system that could read both head and hand gestures using only a video camera. A possible application of this (but not one likely to be developed in the near future) is a computer that can translate American Sign Language (ASL) into speech. A more realistic application would be to allow human hands to manipulate three dimensional objects in synthetic (computer generated/computer controlled) worlds by a system of very specific signs.

8.1.2. Watching More than One User

The current implementation of the head locator does not allow more than one individual to be present in the scene. It is often the case that several people use the same terminal at once, for example, when someone is offering help or giving a demonstration. In simple cases, there is usually one person who is "in charge", and everyone else watches. The problem boils down, then, to deciding which person to look at, so that the computer can respond to her alone.

The dynamics among users sharing a computer terminal are very complicated and beg further study. Certainly, the kinds of predictions the current system makes about the desires of a single user are rendered superfluous in such a situation. In the simplest case, however, it might be possible to mark some of the items on the screen, such as mail icons, as belonging to individual users. If this is done, events that are of interest to each particular user could be targeted specifically to her, by appearing at her gaze point.

8.1.3. Understanding Facial Expressions

A verbal statement (as opposed to a written one) consists of more than its semantic content and syntactic structure. In order to fully understand the speaker, the listener relies on the tone of the speaker's voice (including pitch and inflections), the gestures she uses, and the expressions on her face while the sentence is spoken. Gestures and expressions are not available when people speak with each other on the telephone, so it is sometimes hard to understand the speaker's intent. The problem is further exacerbated in written communication, because vocal inflections are not expressed.[9]

Part of the craft of typography has to do with imitating vocal inflection with the choice of family, size, or boldness of type. Except in literature, plain text is not sufficient to express subtleties of meaning. This is almost always true of transcribed speech. Some work has been done by [Hu 1987] to enhance the appearance of closed-captioned text to match the original inflections of the speaker. As a literary form, electronic mail closely resem-

[9]This problem is more common in notes sent by electronic mail, because the nature of the medium is very informal, that is, people tend to write as they would speak. Since conventions that have been developed to clarify meaning in formal writing are avoided, the intent of the writer is harder to discern. This leads to more misunderstanding among conversers in this particular medium.

bles transcribed speech. Currently, e-mail is shown in a single typeface, so conventions have been established to communicate nuance. For example, in order to indicate a joke, people sometimes type a "smiley face" using characters available on the keyboard (:)). Variations on this icon include a frown (: ( ), and a wink (;)). These icons are usually typed immediately after the ambiguous sentence. It is interesting that the conventions that arose imitate facial expressions, and not some other quality of speech.

It is clear that, if people have a hard time understanding each other without access to the speaker's tone of voice and facial expressions, communication with computers must be severely limited, because they do not at this time have access to the syntactic/semantic elements of speech[10]. It is likely that a system that can understand facial expressions will be available before a system that can understand spoken language. The first such systems will probably recognize simple expressions containing smiles and frowns. More complete systems should recognize expressions of surprise, anger, curiosity, approval and disapproval. In Chapter 3, I showed that the more complex the software, the more subtle the human-computer interaction. Therefore, I believe it is not only possible, but necessary that the kind of work I have discussed in this thesis be applied to the recognition of facial expressions.

8.1.4. Recognizing Individuals

The truly intelligent interface should be able to recognize the individual with whom it is speaking, and make predictions about that user's needs based on prior knowledge of her style. Although the literature is rife with papers about how to program computers recognize human faces, in fact, very little is known about how human beings accomplish this

[10]Research has been conducted at the MIT Media Laboratory [Cahn 1989], to place proper inflection in synthesized speech. The work done by [Hu 1987] to match vocal inflection with text is also relevant.

task. What we do know is that people seem to have the capacity to remember an unlimited number of faces, and that they sometimes have a hard time remembering which personality goes with a particular face. My expectation is that a computer that can recognize individuals will be able to recognize as many faces as a human being can, but will exhibit none of our inconsistency in matching names and faces.

8.2. General Remarks

Good communication should not be limited to the realm of programmers and workstations. Computers are integral parts of many devices on which people habitually rely, and they are becoming pervasive in non-technical life. If they are to be accepted, and if their developers wish to foster an agreeable rapport between machines and the so-called "naive user", the philosophies outlined in this thesis must guide the design process from its inception.

In Fritz Lang's classic film, Metropolis, there is a scene in which a crazed scientist maps a woman's body onto that of a mechanical Doppelganger, to make her a better worker. The analogy to present day human interfaces is plain. It is no longer necessary, nor has it ever been desirable, to mold the man to the machine. Computer technology has reached a level of maturity at which it is possible, even critical, to make machines that serve as perfect channels for ideas, without demanding too much of their users.

Appendix A

Mouse Timing Data

It is unfortunate that almost no study has been made of how people use the mouse in window systems. Card and Moran [Card 1983] did their work before windows were pervasive, and their study of input devices is primarily concerned with perception and motor control, not frequency of use. Some relevant studies have been carried out by [Gaylin 1986] and [Bly 1986], but experiments to determine exactly how much people use the mouse, and under what conditions, do not appear in the literature. My claims about the mouse, therefore, are based on some informal observations of a few expert users. The observations are not subject to scientific scrutiny. However, I believe they indicate a trend.

Ten experienced computer programmers were observed while they worked with a common window system. Their tasks usually consisted of editing, compiling or debugging a program. Each subject was observed for 10 minutes. Both male and female subjects were observed. The subject's hand was considered to be in a neutral position when it was not in contact with the the mouse, but was in contact with some other object (ie, the keyboard or the screen or her own body). I started a stopwatch when a subject's hand left a neutral position, and stopped it when her hand returned to a neutral position. Note that this means that I continued timing as long as a subject's hand was on the mouse. None of the subjects used the mouse as a neutral resting place for her hand. However, three of the subjects were engaged in manipulating a program which was controlled exclusively by the mouse.

The count of the number of windows on the screen reflects the number of windows that were ready to accept keyboard input at the beginning of the observation. This count does not include other items on the screen such as icons or buttons.

The observations are shown in Table 1. below. The average time that a subject's hand was not in a neutral position was 2:52, or almost 30% of the total observation time. To be fair, the high and low data points should be thrown away. Subject [4] worked on a 3-dimensional viewing system that required constant mouse motion. Subject [10] used a text editor exclusively, and did not switch among windows very much at all. Without these data points, the average neutral-position time is 2:36 (~25%).

The number of transitions represents the number of times the subject moved her hand from a neutral position to the mouse, and then back to a neutral position. This number is actually more relevant than the time spent on the mouse. Because of the way window systems are set up, nothing is accomplished while the user's hand is in transition. Therefore, transition time is wasted time. The average number of transitions is 21.3, in a 10 minute period. Throwing away subjects [4] and [10], as above, the average number of transitions is approximately the same (21.5).

Subject	# of Transitions	Non Keyboard/ Neutral Time	# of Windows on Screen
[1]	15	1:49	5
[2]	13	4:19	10
[3]	55	3:40	4
[4]	24	6:49	3
[5]	14	3:43	2
[6]	12	1:04	3
[7]	14	2:58	4
[8]	19	2:12	4
[9]	30	1:12	6
[10]	17	0:57	5

Mouse Timing Data

One would expect to find a correlation between the number of transitions and the number of windows on the screen. This was not observed, but I believe that is because of the small number of subjects involved. A more rigorous study would require more subjects, in order to encompass a wider population.

Appendix B.

Technical Specifications

The head server is implemented on a Sun 3/60, which runs SunOS 4.0, a modified version of Berkeley Unix version 4.2. The Sun is a diskless node on a local area net running NFS (Networked File System). The Data Cube frame buffer allows up to 4 simultaneous inputs (R,G,B,Y). An NTSC video signal is fed into the Y channel for digitization. The following chart describes the hardware components in more detail. The resulting digital image is 768x512 pixels, displayed using an 8 bit color lookup table. Live video is captured from an analog (tube) camera. Automatic Gain Control was turned off for this application.

Minicomputer	**Sun 3/60**
Central Processor	68020
Floating Point Processor	68881
Network Interface	Ethernet
Display Screen Size	19" diagonal
Frame Buffer	**Data Cube Model BG123**
Number of Inputs	4
Depth	8 bits (256 color table)
Digital Resolution	768x512
Digitized Video Resolution	768x480
Camera	**Panasonic WV-1550**
Video Element	2/3" Vidicon tube
Automatic Gain Control (AGC)	Off
Sensitivity	0.3 lux
Output Signal	NTSC
Lens	**Computar**
Apeture	variable 1:1.8
Focal Length	25mm

Appendix C

Client-Server Protocol

Client Request Types

There are currently 5 Client request types and 3 Server Acknowledgement request types. The client may send any of the following to the server, in any order, as long as the last request sent is a suicide request:

ENQ_FULL_ANALYSIS

Requests a full description of the distance and attitude of the user's head, relative to the screen. If it is successful in processing the incoming image, the server returns the following items in the Request packet reply:

- The distance from the screen to the tip of the user's nose (XDist)
- The elevation from the middle of the screen of the user's nose (YDist)
- The east/west angle of gaze (pan)
- The north/south angle of gaze (tilt)

ENQ_HEAD_ONLY

Requests that the server send only the XDist and YDist values, which report the distances from the front of the computer screen to the tip of the user's nose, and from the top of the screen to the top of the user's head. This is used in clients that do not need to know the angle of the user's head, but do need to know the user's distance from the screen.

ENQ_SET_ACTIVEWINDOW

This is the only request that can send data to the server. The data portion contains the name of an X Window (an X Window name is an unique window identification number set by X11.) The server stores the name of the active window in an internal register where it can be read by other clients.

ENQ_GET_ACTIVEWINDOW

Requests that the server send the name of the active window stored in the internal register reserved for that purpose.

ENQ_SUICIDE

Issued to notify the server that the is about to close the connection to the head server. The connection is actually broken by a call to *CloseServerConnection()*.

Server Reply Request Types

After the client has sent a packet containing a Client Request, it must wait until the Server processes the request. The server returns a packet that contains one of the following request types.

ACK

Indicates that the request was processed successfully, without any intervening errors. Usually, the client will receive an ACK in response to an ENQ_FULL_ANALYSIS request, in which case the data portion of the request packet will contain information about the head location. The information is a formatted ASCII string that would print as four floating point numbers. Nota Bene: the data is not four floating point numbers, it is a string such as: "254.00 123.93 22.50 16.42". This was done in order to maintain compatibility when the client and server are on different machines, with different internal floating point formats.

NACK

Indicates that some error occured while processing the request. The data in the data packet is unpredictable.

NOT_PRESENT

Indicates that no user was present in front of the video camera at the time of the request. There is no reliable data associated with this reply.

Acknowledgements

There are many people to whom I am sincerely grateful for helping me complete my studies and this thesis. I would like to single out:

Walter Bender, my thesis advisor, who'se creativity, enthusiasm and energetic skepticism I will always admire.

My family: Mom, Dad, Alan and Josh, for your unconditional love. Having you just around the corner (and sometimes in my classroom) has been a special gift.

Jennifer, with tender affection, who sadly will never be the same.

And, finally, IBM, for letting me play with all those nifty toys.

References

Apollinaire, Guillaume. 1980. Calligrammes. Translated by Anne Hyde Greet. Los Angeles: University of California Press.

Bæker, R. M., and William .A.S. Buxton. "A historical and intellectual perspective." Readings in human-computer interaction: A multidisciplinary approach. Ed. R.M. Bæker and W.A.S. Buxton. Los Altos, California: Kaufman, 1987.

Bly, S. A., and J. K. Rosenberg. "A comparison of tiled and overlapping windows." Computers and Human Interaction - CHI'86 : 101-106.

Bolt, Richard. "Put That There: Voice and Gesture at the Graphics Interface." ACM CG 14.3 (1980): 262-270.

Bush, Vannevar. 1945. "As We May Think." The Atlantic Monthly, July, 1945, v171 101-108.

Card, Stuart K., Thomas P. Moran, and Allen Newell. 1983. The Psychology of Human-Computer Interaction. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Carroll, John M., and Judith Reitman Olson. "Mental Models in Human-Computer Interaction." Handbook of Human-Computer Interaction. Ed. Martin Helander. Amsterdam: North-Holland, 1988.

Chignell, M. H., and P.A. Hancock. "Intelligent Interface Design." Handbook of Human-Computer Interaction. Ed. Martin Helander. Amsterdam: North-Holland, 1988.

Donath, Judith S. 1983. The Electronic Newstand: Design of an Intelligent Interface to a Variety of News Sources in Several Media. Master's Thesis, Massachusetts Institute of Technology.

Elkerton, Jay. "Online Aiding for Human-Computer Interfaces." Handbook of Human-Computer Interaction. Ed. M. Helander. Amsterdam: North-Holland, 1988.

Fisher, Scott Stevens. 1981. Viewpoint Dependent Imaging: An Interactive Stereoscopic Display. Master's Thesis, Massachusetts Institute of Technology.

Foley, James D. 1987. "Interfaces for Advanced Computing." Scientific American, October, 1987, 257 (4):127-135.

Gaylin, K. B. "How are windows used? Some notes on creating an empirically-based windowing benchmark task." Human Factors in Computing Systems - CHI'86 : 96-101.

Ginsberg, Carol Marsha. 1983. Human Body Motion as Input to an Animated Graphical Display. Master's Thesis, Massachusetts Institute of Technology.

Gulsen, Denis R. 1988. User Presence Detector. Bachelor's Thesis, Massachusetts Institute of Technology.

Hochberg, Julian. "The Semantics of Attention." Attention: Contemporary Theory and Analysis. Ed. David I. Mostofsky. 1 vols. Century Psychology Series. New York: Meredith Corporation, 1970. 1: 447.

Hu, Antonio C. 1987. Automatic Emphasis Detection in Fluent Speech with Transcription. Bachelor's Thesis, Massachusetts Institute of Technology.

Jacob, Robert. "Eye Tracking." SIGGRAPH Conference Notes 1 : 65 to 145.

Jacob, Robert J. K. 1989. What You Look At Is What You Get: The Use of Eye Movements in Human-Computer Interaction Techniques. Human-Computer Interaction Lab, Naval Research Laboratory Washington, D.C.

Jenkins, Francis A., and Harvey E. White. 1950. Fundamentals of Optics. 2nd ed. New York: McGraw-Hill.

Licklider, J.C.R. "Man-Computer Symbiosis." IRE Trans. Human Factors 1.1 (1960): 4-11.

Mase, Kenji, Yasuhito Suenaga, and Tsaka-aki Akimoto. "Head Reader." Proc. IEEE SMC. October (1987).

Mase, Kenji, Yasuhiko Watanabe, and Yasuhito Suenaga. "A Realtime Head Motion Detection System." SPIE SR3D 1260.1 (1990): 262-269.

Maxwell, D. R. 1983. Graphical Marionette: A Modern-Day Pinocchio. Master's Thesis, Massachusetts Institute of Technology.

Mollitor, Robert Charles. 1990. Eloquent Scenery: A Study of Peripheral Visual Communication. Master's Thesis, Massachusetts Institute of Technology.

Pedretti, Carlo, ed. The Literary Works of Leonardo Da Vinci. Berkeley: University of California Press, 1977. 1:

Rabb, F. H., and et. al. "Magnetic position and orientation tracking system." IEEE trans. Aerosp. & Electron Syst. 15.5 (1979):

Rissland, Edwina L. "Ingredients of intelligent interfaces." Readings in Human-Computer Interaction: A Multidisciplinary Approach. Ed. Ronald M. Bæker and William A. S. Buxton. San Mateo, California: Morgan Kaufmann Publishers, Inc., 1987. 703-709.

Salomon, Gitta B. 1983. Design and Implementation of An Electronic Special Interest Magazine. Master's Thesis, Massachusetts Institute of Technology.

Schmandt, Chris, Mark S. Ackerman, and Debby Hindus. 1989. Augmenting a Window Manager with Speech Input. Massachusetts Institute of Technology.

Sturman, David J., David Zeltzer, and Steve Peiper. "Hands-on Interaction With Virtual Environments." UIST '89 Symposium on User Interface Software and Technology :

Sutherland, Ivan E. "A head-mounted three dimensional display." AFIPS Fall Joint Computer Conference, December 9-11, 1968, 33 part 1 : 757-764.

Wærn, Yvonne. 1989. Cognitive Aspects of Computer Supported Tasks. Chichester: John Wiley and Sons.

Zimmerman, Thomas G. et al. "A Hand Gesture Interface Device." ACM CHI+GI .Special Issue (1987): 189-192.

This Thesis was typeset in Adobe Garamond 3, under lisence from the International Typeface Corporation (ITC). Garamond was commissioned from the Imprimerie Nationale, the French royal typographic monopoly, in 1540 by François 1st. It has been a favorite of book designers for many centuries. In 1900 Ambroise Vollard, a publisher and book designer who was influential in the *livre-art* movement of the late 19th century, described Garamond as "That magnificent type...the italics of which seem to me expressly designed to print the work of a poet."